Instruction Manual

Fuji Electric
Innovating Energy Technology

FRENIC-ACE

OPC-PRT
Multiprotocol Ethernet Interface

⚠CAUTION

Thank you for purchasing the OPC-PRT Multiprotocol Ethernet Interface.

- This product is designed to connect the FRENIC-Ace series of inverters to Ethernet communication networks. Please read this instruction manual thoroughly in order to become familiar with the proper interface handling, installation and usage procedures.
- Improper handling may inhibit correct operation or cause premature interface failure.
- Please deliver this instruction manual to the end user of the interface, and retain it in an accessible location.
- For inverter usage instructions, please refer to the applicable inverter instruction manual.

July 31, 2015
Part #10949

© 2015 Fuji Electric

OPC-PRT Multiprotocol Ethernet Interface Instruction Manual

Part Number 10949
Printed in U.S.A.
©2015 Fuji Electric.
All rights reserved

Fuji Electric reserves the right to make changes and improvements to its products without providing notice.

Notice to Users

PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS. Life-support devices or systems are devices or systems intended to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs may always be present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

Preface

This instruction manual has been prepared to help you connect your FRENIC-Ace inverter to Industrial Ethernet networks using the OPC-PRT Multiprotocol Ethernet interface card. This instruction manual does not contain inverter usage instructions. Please refer to this instruction manual in conjunction with the applicable inverter instruction manual in order to become familiar with the proper handling, installation and operation of this product. Improper handling or installation procedures may result in incorrect operation or premature product failure.

Related Publications

Listed below are publications that are necessary for reference in conjunction with this instruction manual.

- **RS-485 User's Manual (24A7-E-0082)**
- **FRENIC-Ace Instruction Manual (INR-SI47-1733a-E)**
- **FRENIC-Ace User's Manual (24A7-E-0043E)**

These documents are subject to change without notice. Please be sure to refer to the most recent available versions.

Safety precautions

Please read this instruction manual thoroughly prior to proceeding with installation, connections, operation, or maintenance and inspection. Additionally, ensure that all aspects of the system are fully understood, and familiarize yourself with all safety information and precautions before operating the inverter.

Safety precautions in this instruction manual are classified into the following two categories:

⚠WARNING	Failure to heed the information indicated by this symbol may lead to dangerous conditions, possibly resulting in death or serious bodily injuries.
⚠CAUTION	Failure to heed the information indicated by this symbol may lead to dangerous conditions, possibly resulting in minor or light bodily injuries and/or substantial property damage.

Failure to heed the information contained under the CAUTION title can also result in serious consequences. These safety precautions are of utmost importance and must be observed at all times.

Installation and Wiring

⚠WARNING

- To avoid electrical shock, remove all power from the inverter and wait at least five minutes prior to starting installation. Additionally, confirm that the DC link bus voltage as measured between the P (+) and N (-) terminals is less than 25 VDC.
- Installation should be performed only by qualified personnel.
- To avoid electrical shock, do not operate the inverter with the front cover or wiring cover removed, as accidental contact with exposed high-voltage terminals and internal components may occur.
- To prevent explosions or similar damage, ensure that all cables are properly connected to the correct terminals, and observe all wiring polarity indicators.

⚠CAUTION

- Do not install or operate the interface if it is damaged or has parts missing.
- Prevent conductive items such as screws and metal fragments, or flammable substances such as oil, lint, paper fibers and sawdust from entering the inverter and interface card enclosure.
- Incorrect handling during installation or removal may cause equipment failure.
- Do not subject the cables to scratches, excessive stress, heavy loads or pinching.
- To prevent damage due to electrostatic discharge, always touch a grounded piece of metal prior to touching any equipment.
- Do not stand on or rest heavy objects on the equipment.
- To prevent burns from hot components, do not touch the inverter while power is on, or for some time after power is removed.
- Electrical noise may be emitted from the inverter, motor and wires. Always implement appropriate countermeasures to prevent nearby sensors and devices from malfunctioning due to such noise.

Operation

⚠WARNING

- To avoid electrical shock, do not open the front cover of the inverter while power is on or while the inverter is running.
- To avoid electrical shock, do not operate switches with wet hands.
- If the inverter’s function codes are incorrectly configured, or configured without adequate understanding of the appropriate inverter Instruction Manual and User's Manual, the motor may rotate with a torque or at a speed not permitted for the machine. Confirm the settings of all function codes prior to running the inverter.

Maintenance, inspection, and parts replacement

⚠WARNING

- To avoid electrical shock, remove all power from the inverter and wait at least five minutes prior to starting inspection. Additionally, confirm that the DC link bus voltage as measured between the P (+) and N (-) terminals is less than 25 VDC.
- Maintenance, inspection, and parts replacement should be performed only by qualified personnel.
- Remove all watches, rings and other metallic objects prior to starting work.
- To avoid electrical shock or other injuries, always use insulated tools.

Disposal

⚠CAUTION

- Contact the local or state environmental agency in your area for details on the disposal of electrical components and packaging.

Other

⚠WARNING

- Do not attempt to modify the equipment: doing so may cause electrical shock or injuries.
- For clarity purposes, illustrations in this manual may be drawn with covers or safety guards removed. Ensure all covers and safety guards are properly installed prior to starting operation.
- Do not perform hi-pot tests on the equipment.
- Performing a data initialization (function code H03) may reset all inverter function codes to their factory default settings. After performing this operation, remember to reenter any custom function code values prior to starting operation.

Icons

The following icons are used throughout this manual:

Indicates information which, if not heeded, can result in the product not operating to full efficiency, as well as information concerning incorrect operations and settings which may result in accidents.

Indicates information that can prove handy when performing certain settings or operations.

Indicates a reference to more detailed information.

– TABLE OF CONTENTS –

1 PRE-OPERATION INSTRUCTIONS 8

1.1 Product Overview 8

1.2 Features and Specifications 8

1.3 Unpacking and Product Confirmation 13

1.3.1 Shipment Confirmation 13

1.3.2 Component Overview 14

1.4 LED Indicators 15

1.4.1 Network Status LED 15

1.4.2 Module Status LED 15

1.4.3 Ethernet Link/Activity LEDs 15

2 INSTALLATION 16

2.1 Pre-Installation Instructions 16

2.2 Installation Procedure 16

3 INVERTER FUNCTION CODE SETTINGS 21

3.1 Inverter Control-Related Settings 21

3.2 Inverter Reaction to Network Timeout Conditions 22

4 FUNCTION CODE NUMBERING AND BEHAVIOR 23

4.1 Register Numbers 23

4.2 Scanned Function Codes 27

4.3 Commonly Used Function Codes 27

5 FUJI CONFIGURATION STUDIO 29

5.1 Overview 29

5.2 General Object Editing Activities 31

5.3 Ethernet Settings 32

5.3.1 Authentication 32

5.3.2 Network Configuration 32

5.4 Internal Logic Settings 32

5.4.1 Fail-safe Values 32

5.4.2 Fail-safe Example 33

5.5 Discovery over Ethernet 34

5.6 Manage Device Parameters 34

5.7 Backup and Restore Parameters 35

5.8 Restore Factory Settings 36

5.9 Help 36

6 EMBEDDED WEB SERVER 37

6.1 Overview 37

6.2 Monitor Tab 37

6.2.1 Information Window 37

6.2.2 Function Code Group Selection List 38

6.2.3 Function Code List 39

6.2.4 Function Code List Filter 39

6.2.5 Radix Selection ... 40
6.3 Dashboard Tab ... 41
6.3.1 Information Window ... 41
6.3.2 Virtual Keypad ... 42
6.3.3 Gauge Window Navigation ... 43
6.3.4 Gauge Window Configuration ... 43
6.3.5 Submitting Changes ... 46
6.4 Customizing the Embedded Web Server ... 47
6.4.1 Customization Overview ... 47
6.4.2 XTPro Overview ... 47
6.4.3 XTPro Web Browser-Based Implementation ... 48
6.4.4 XTPro HMI-Based Implementation ... 49
6.4.5 XTPro Supported Commands ... 49
7 FILE SYSTEM & FIRMWARE ... 50
7.1 File System ... 50
7.1.1 Overview ... 50
7.1.2 USB with Windows Explorer ... 50
7.1.3 FTP with Windows Explorer ... 51
7.1.4 Loading New Web Server Content ... 51
7.2 Firmware ... 52
7.2.1 Overview ... 52
7.2.2 Update Procedure ... 52
8 PROTOCOL-SPECIFIC INFORMATION ... 53
8.1 Modbus/TCP ... 53
8.1.1 Overview ... 53
8.1.2 Holding & Input Registers ... 53
8.1.3 Coil & Discrete Input Mappings ... 53
8.1.4 Connection Timeout Options ... 54
8.1.5 Node Settings ... 55
8.1.6 Holding/Input Register Remap Settings ... 55
8.2 EtherNet/IP ... 56
8.2.1 Overview ... 56
8.2.2 Server Settings ... 56
8.2.3 Connection Timeout Options ... 57
8.2.4 Generic Class 1 I/O Produced and Consumed Data Settings ... 57
8.2.5 Generic Class 1 (I/O) Connection Access ... 58
8.2.6 AC/DC Drive Profile Class 1 (I/O) Connection Access ... 58
8.2.7 Explicit Messaging Via Data Table Read/Write Services ... 60
8.2.8 Inverter Function Code Access Tag Format ... 61
8.2.9 ControlLogix Examples: Setup ... 61
8.2.10 ControlLogix Example: I/O Messaging ... 62
8.2.11 ControlLogix Example: Generic Default I/O Add-On Instruction ... 65
8.2.12 ControlLogix Example: AC/DC Drive Profile Add-On Instruction ... 67
8.2.13 ControlLogix Example: Read a Block of Function Codes ... 70
8.2.14 ControlLogix Example: Read a Single Function Code ... 74
8.2.15 ControlLogix Example: Multiple MSG Instructions ... 74
8.2.16 ControlLogix Example: Reading and Writing ... 75
8.3 Allen Bradley CSP (PCCC) ... 76
8.3.1 Overview ... 76
8.3.2 Explicit Messaging Via Typed Read/Write Services ... 76

8.3.3 Inverter Function Code File Number Offset Format ... 76
8.3.4 SLC-5/05 Example: Read a Block of Function Codes ... 78
8.3.5 SLC-5/05 Example: Read a Single Function Code ... 83
8.3.6 SLC-5/05 Example: Multiple MSG Instructions ... 84
8.3.7 SLC-5/05 Example: Reading and Writing ... 85
8.4 BACnet/IP ... 86
8.4.1 Protocol Implementation Conformance Statement ... 86
8.4.2 Default Supported Objects ... 90
8.4.3 Default Supported Object Details ... 92
8.4.4 Server Settings ... 93
8.4.5 Node Settings ... 93
8.4.6 Device Object Settings ... 93
8.4.7 BACnet Object Settings ... 93
8.4.8 Analog Input Object Settings ... 93
8.4.9 Analog Output Object Settings ... 93
8.4.10 Analog Value Object Settings ... 94
8.4.11 Binary Input Object Settings ... 94
8.4.12 Binary Output Object Settings ... 95
8.4.13 Binary Value Object Settings ... 96
8.4.14 Multi-state Input Object Settings ... 97
8.4.15 Multi-state Output Object Settings ... 97
8.4.16 Multi-state Value Object Settings ... 97
8.5 PROFINET IO ... 98
8.5.1 Overview ... 98
8.5.2 Device Settings ... 98
8.5.3 Connection Timeout Options ... 98
8.5.4 Cyclic I/O Produced and Consumed Data Access Settings ... 98
8.5.5 PROFIdrive Profile ... 99
8.5.6 Acyclic Data Access ... 103
8.5.7 STEP 7 Hardware Configuration Example ... 103
9 TROUBLESHOOTING ... 107

1 PRE-OPERATION INSTRUCTIONS

1.1 Product Overview

The OPC-PRT Multiprotocol Ethernet interface allows information to be transferred seamlessly between a FRENIC-Ace inverter and several Ethernet-based fieldbus networks with minimal configuration requirements. The interface installs directly onto the inverter, and presents two RJ-45 jacks with an embedded 10/100BaseT Ethernet switch for connection to the Ethernet network. In addition to the supported fieldbus protocols, the interface also hosts a fully-customizable embedded web server, which provides access to inverter information via a standard web browser for remote monitoring and control.

Before using the interface, please familiarize yourself with the product and be sure to thoroughly read the instructions and precautions contained in this manual. In addition, please make sure that this instruction manual is delivered to the end user of the interface, and keep this instruction manual in a safe place for future reference or unit inspection.

Note that different interface firmware versions may provide varying levels of support for the various protocols. When using this manual, therefore, always keep in mind the release date of the firmware version running on your interface as it must correspond to this manual's respective release date in order for all documented aspects to apply.

Supported Protocols

The interface currently provides server support for the following fieldbus protocols:

- Modbus/TCP Server
- EtherNet/IP Server
- Allen Bradley CSP Server (also known as "PCCC" and "AB Ethernet")
- BACnet/IP Server
- PROFINET IO Device (MRP client)

1.2 Features and Specifications

Table 1: Features

Item	Description
Simultaneous Protocols	Supports all standard unmodified Ethernet (SUE) protocols simultaneously
Fuji Configuration Studio	Graphical user interface for discovery, configuration, and firmware update
WEB Server (HTTP)	Access all parameters, dashboard with gauges, customizable with XTPro
Communication Loss Detection	Configurable actions for "fail-safe" conditions
Field Upgradeable	Firmware updates automatically handled by the studio
Parameter Management	Advanced management of parameter access and scan priority
Parameter Backup and Restore	Drive cloning

Table 2: General Hardware Specifications

Item	Description
Power Supply	Directly powered by the inverter
LED Indicators	Module Status, Network Status, 2 x Ethernet Link/Activity
USB Port	USB 2.0, mini-B 5-pin

Table 3: Ethernet Hardware Specifications

Item	Description
Number of Ports	2 (internal switch)
Standard	IEEE 802.3 10/100BaseT Ethernet compliant
Communication Speed and Duplex	10Mbps half/full, 100Mbps half/full (auto sense optimal speed and duplex)
Connector Type	RJ-45 Shielded
Auto MDI-X	Yes (supports all straight-through and cross-over cables)
Cable Type	CAT5-type 8-conductor UTP patch cables
Cable Length	100m per segment max
Topologies	Star/Tree, Linear/Bus/Daisy-chain, Ring (MRP)

Table 4: Modbus/TCP Specifications

Item	Description
Conformance Class	Class 0, Class 1 (partial), Class 2 (partial)
Read Function Codes	Read coils (1), Read input status (2), Read multiple registers (3), Read input registers (4)
Write Function Codes	Write coil (5), Write single register (6), Force multiple coils (15), Write multiple registers (16)
Number of Connections	8
Max Read Register Size	125 registers
Max Write Register Size	123 registers
Register Data Type	16-bit integer
Unit (slave) ID	Ignored, echoed in response
TCP Port	502
Response Time	Min 160us, Typically less than 1ms

Table 5: EtherNet/IP Specifications

Item	Description
Conformance Tested	ODVA EtherNet/IP Conformance Test Software Version CT12
Product Type Code	2 (AC Drive)
AC/DC Drive Profile	Yes
UCMM	Yes
Class 1 (Implicit I/O) Messaging	Yes
Class 3 (Explicit) Messaging	Yes
Number of Connections	16 (Total for both Class 1 and Class 3)
RPI	Min 1ms
I/O Input Size	Max 32 input words, user configurable
I/O Output Size	Max 32 output words, user configurable
Generic (User Configurable) Assembly Instances	100 (input) and 150 (output)
AC/DC Drive Profile Assembly Instances	20 (input) and 70 (output), 21 (input) and 71 (output)
Data Table Read/Write	Yes
Class 1 UDP Port	2222 (0x08AE)
Explicit Messaging Port	44818 (0xAF12)
Explicit Messaging Response Time	Min 160us, Typically less than 1ms

Table 6: Allen Bradley CSP (PCCC) Specifications

Item	Description
Read Services	PLC5 Read (DF1 protocol typed read, 0x68)
Write Services	PLC5 Write (DF1 protocol typed write, 0x67)
Data Type	16-bit Integer
File Type	N (Integer)
Logical ASCII Addressing	Yes
Logical Binary Addressing	Yes
Max Read Size	240 bytes (120 16-bit Integers)
Max Write Size	240 bytes (120 16-bit Integers)

Table 7: BACnet/IP Specifications

Item	Description
BACnet IP	Annex J
Protocol Revision	2
Standard Device Profile (Annex L)	BACnet Application Specific Controller (B-ASC)
BACnet Interoperability Building Blocks (BIBB)	ReadProperty-B (DS-RP-B), ReadPropertyMultiple-B (DS-RPM-B), WriteProperty-B (DW-WP-B), Dynamic Device Binding-B (DM-DDB-B), Dynamic object Binding-B (DM-DOB-B)
Segmentation	No
Max APDU Length	1444 bytes
Character Sets	ANSI X3.4
Object Types	Analog Output, Analog Input, Analog Value, Binary Output, Binary Input, Binary Value, Multi-state Output, Multi-state Input, Multi-state Value
Priority Array	Yes
UDP Port	47808 (0xBAC0, configurable)
Response Time	Min 160us, Typical less than 1ms

Table 8: PROFINET Specifications

Item	Description
Protocol Level	RT (real-time)
RT Conformance Class	Class A
I/O Cycle Time	Min 1ms
I/O Input Size	Max 32 input words, user configurable
I/O Output Size	Max 32 output words, user configurable
MRP	Media Redundancy Protocol Client
DCP	Discovery, set station name, set IP address
LLDP	Yes
I&M	I&M0
Alarms	Plug, Pull
Number of Controllers	Allows access to only 1 controller

Table 9: Applicable Inverters

Series	Type	Capacity	ROM version
FRENIC-Ace	FRN□□□E2□-□□	All capacities	0300 or higher

Table 10: Environmental Specifications

Item	Description
Operating Environment	Indoors, less than 1000m above sea level, do not expose to direct sunlight or corrosive / explosive gasses
Operating Temperature	-10 ~ +50°C (+14 ~ +122°F)
Storage Temperature	-40 ~ +85°C (-40 ~ +185°F)
Relative Humidity	20% ~ 90% (without condensation)
Vibration	5.9m/s^2 (0.6G) or less (10 ~ 55Hz)
Cooling Method	Self-cooled
RoHS (Lead free)	Yes

1.3 Unpacking and Product Confirmation

1.3.1 Shipment Confirmation

Check the enclosed items. Confirm that the correct quantity of each item was received, and that no damage occurred during shipment.

- OPC-PRT interface board with spacer and captive M3 x 12mm screw in lower-right corner (refer to Figure 1).
- One separate M3 x 6mm mounting screw (see Figure 2).
- Type A male to mini-B male USB interface cable (see Figure 3).

Figure 1: OPC-PRT Interface Board

Figure 2: M3 x 6mm Mounting Screw

Figure 3: USB Interface Cable

1.3.2 Component Overview

Figure 4 provides an overview of the important interface card components.

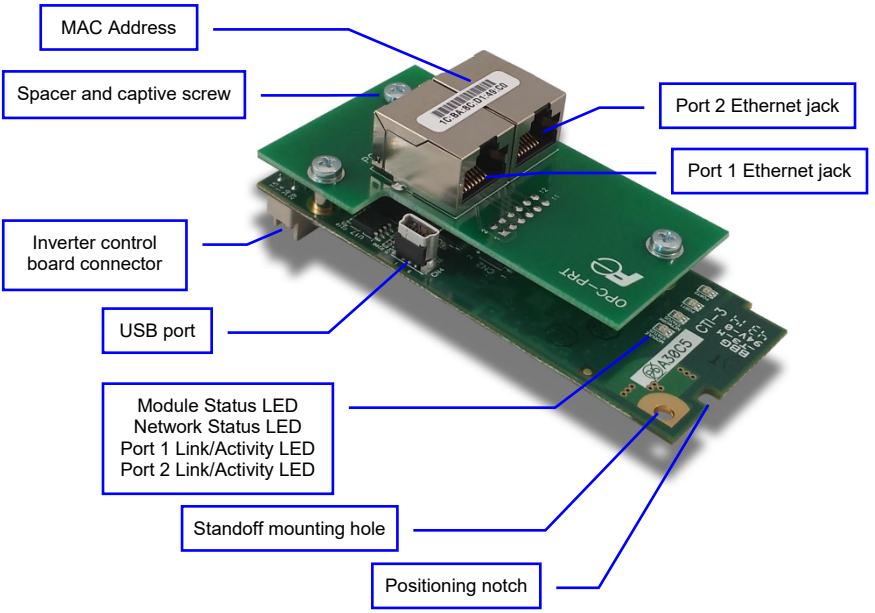


Figure 4: OPC-PRT Component Overview

Positioning Notch

Aligns with the positioning key on the inverter chassis to ensure that the interface card is installed into the correct communication port (refer to section 2.2).

Port 1 and Port 2 Ethernet Jacks

Either jack can freely be used in star topology networks (with external switch). In linear topologies, a series of cards can be connected together by daisy-chaining one of the ports to the next inverter in line. In ring topologies, MRP (Media Redundancy Protocol) must be supported by all devices on the network.

Standoff Mounting Hardware

The provided M3 x 12mm and M3 x 6mm screws are used to secure the card to the standoffs located on the inverter's control board. Refer to section 2.2.

Inverter Control Board Connector

Attaches to the inverter's connector board, which may vary depending on the inverter model.

USB Port

USB 2.0 port with mini-B connector. Used to access the card via the Fuji Configuration Studio (refer to section 5) and as a USB flash drive (refer to section 7.1).

Module Status and Network Status LEDs

These LEDs indicate the current status of the interface card and protocols in use. Refer to section 1.4.

Ethernet Link and Activity LEDs

One set of LEDs are provided for each Ethernet port. These LEDs provide insight into the Ethernet network's status and activity. Refer to section 1.4.

1.4 LED Indicators

1.4.1 Network Status LED

LED Activity	Status	Note
Off	Device Off	The inverter power is off
Green Blink / Red Blink	Startup	Startup blink sequence
Green Blink	No Connection	EtherNet/IP connection is not established
Green Off	No Connection	PROFINET connection is not established
Green On	Connection Established	EtherNet/IP or PROFINET connection is established

1.4.2 Module Status LED

LED Activity	Status	Note
Off	Device Off	The inverter power is off
Green Blink / Red Blink	Startup	Startup blink sequence
Green On	Device On	Normal status
Green Blink	Discovery identification	PROFINET discovery and identification (DCP)
Red Blink	Error Code	Record the error code sequence and contact technical support

1.4.3 Ethernet Link/Activity LEDs

LED Activity	Status	Note
Green On	Link	A valid Ethernet link exists: communication is possible on this port
Green Off	No Link	A valid Ethernet link does not exist: communication is not possible on this port
Red Blink	Activity	Indicates when a packet is transmitted or received on this port

2 INSTALLATION

2.1 Pre-Installation Instructions

⚠WARNING

- To avoid electrical shock, remove all power from the inverter and wait at least five minutes prior to starting installation. Additionally, confirm that the DC link bus voltage as measured between the P (+) and N (-) terminals is less than 25 VDC.
- Installation should be performed only by qualified personnel.
- To avoid electrical shock, do not operate the inverter with the front cover or wiring cover removed, as accidental contact with exposed high-voltage terminals and internal components may occur.
- To prevent explosions or similar damage, ensure that all cables are properly connected to the correct terminals, and observe all wiring polarity indicators.
- Only one additional option card may be used when the OPC-PRT is installed in the inverter. If two additional option cards are required, please consult with the factory first to confirm compatibility.

2.2 Installation Procedure

Note Before installing the interface card, perform all wiring for the main circuit terminals and control circuit terminals.

1. Remove the front cover from the inverter to expose the control printed circuit board (control PCB). Install the interface card according to the inverter capacity as shown in Figure 5, Figure 6, or Figure 7. Otherwise, refer to the FRENIC-ACE Instruction Manual or contact Fuji for the appropriate installation instructions.

   📖 To remove the front cover, refer to the FRENIC-Ace Instruction Manual, Section 2.2.

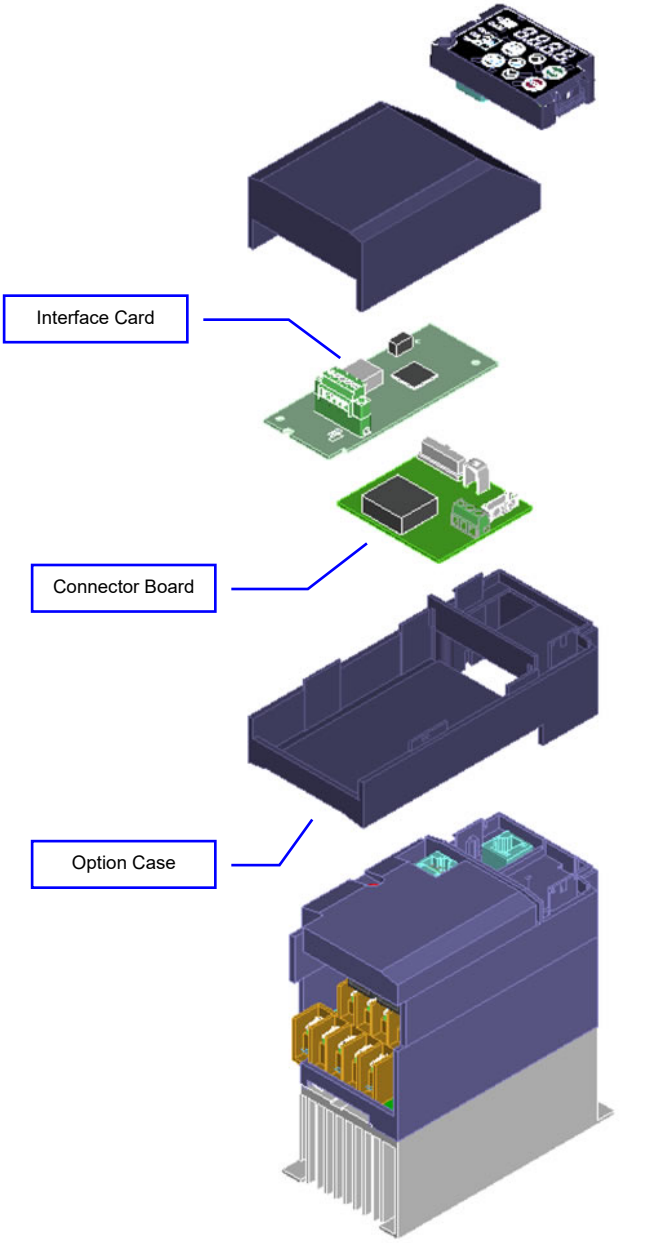


Figure 5: Installation for 15 kW and Smaller Inverters

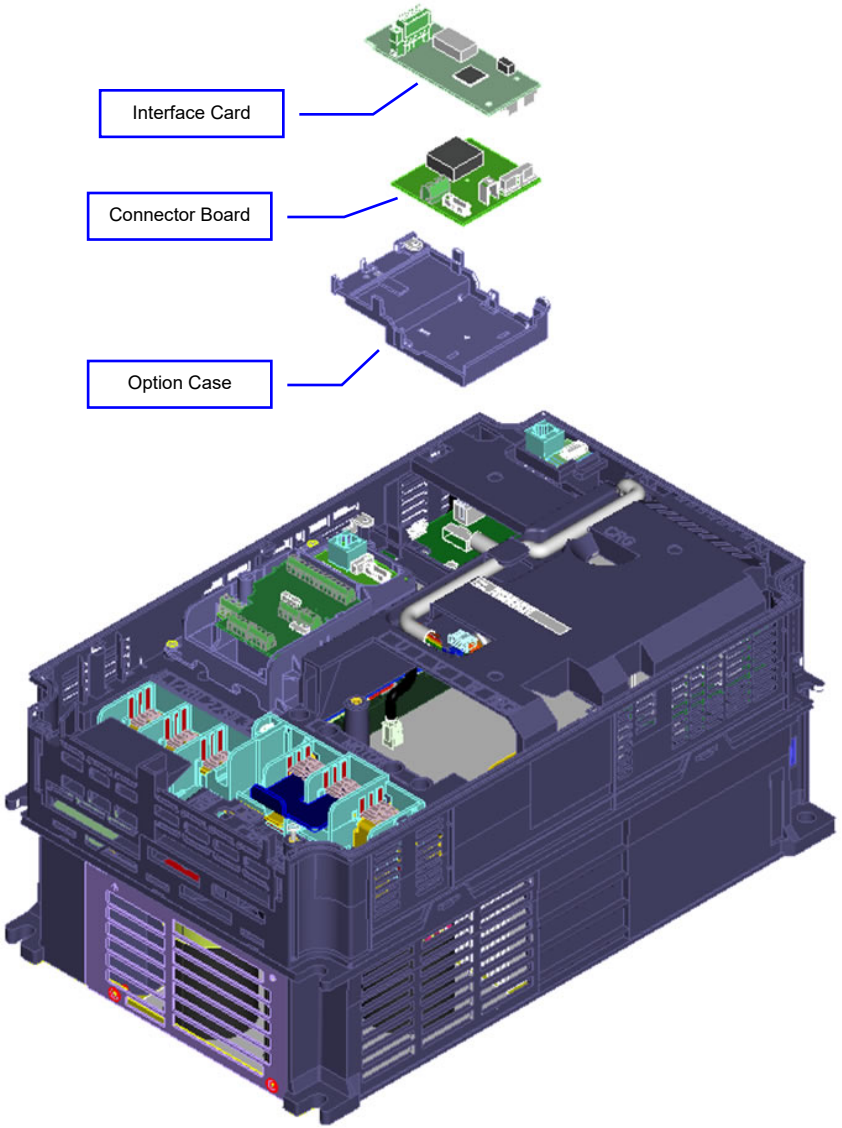


Figure 6: Installation for 18.5 kW to 22 kW Inverters

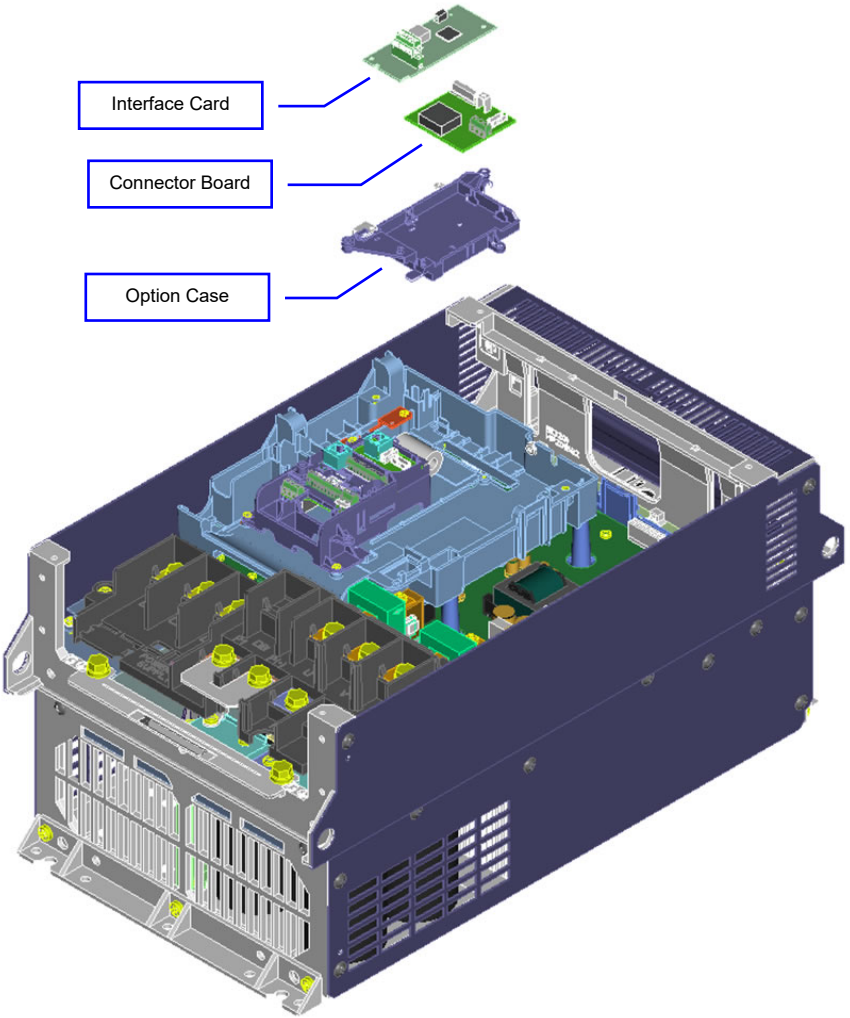


Figure 7: Installation for 30 kW and Larger Inverters

2. Engage connector CN1 (on the back of the interface card) into the connector on the connector board. Ensure that the connectors are fully engaged.

Note

Ensure that the interface card is fully aligned and seated into the communication port. Failure to do so may lead to insufficient connector insertion and result in contact failure.

3. Secure the interface card to the connector board PCB by first tightening the captive M3 x 12mm screw into the inverter standoff located at the lower-right hand corner of the interface card. Next, install and tighten the included M3 x 6mm screw into the standoff mounting hole located at the upper-left hand corner of the interface card.

4. Connect the network cables as necessary. Insert the Ethernet cables into the Ethernet jacks, making sure that they are fully seated. Ensure that the cables are routed in such a way that they

will not be pinched and are not located near any power-carrying wiring, such as the inverter's input power or motor wires.

5. Reinstall all covers removed in step 1. Take a moment to confirm that the Ethernet cables are not being pinched and are not routed near any power-carrying wiring.

   For reinstallation instructions, refer to the FRENIC-Ace Instruction Manual, Section 2.2.

3 INVERTER FUNCTION CODE SETTINGS

Depending on the desired operation of the overall application, the inverter function codes listed in Table 11 are important for proper operation of the end-to-end communication system. Although there may be many other function codes that will require configuration for your specific application, it is important to understand the manner in which the following function codes will impact successful control of the inverter.

For further details regarding these function codes, please refer to the **FRENIC-Ace Instruction Manual (INR-SI47-1733a-E)**, Chapter 5 "FUNCTION CODES", **FRENIC-Ace User's Manual (24A7-E-0043E)**, "y codes: Link Functions", and **RS-485 User's Manual (24A7-E-0082)**, Chapter 5, Section 5.2 "Data Formats."

Table 11: Function Code Settings Overview

Code	Name	Setting Range	Required Value
Y98	Bus Link Function (Mode Selection)	0 to 3	3

3.1 Inverter Control-Related Settings

The following function codes relate to whether or not the inverter is to be controlled (command word and/or frequency command) from the network, or whether the inverter will be locally-controlled (and therefore only monitored and/or configured via the network.)

Bus Link Function (Mode Selection) (y98)

If the inverter is to be controlled from the network, then set the value of y98 to 3 (fieldbus option). A setting of 3 for y98 may also be appropriate even if H30 is configured for an alternate (local) control scheme.

When the inverter is controlled from the network, a selection of reference commands (S## function codes as defined in Table 12) are available for controlling the inverter's speed. If multiple reference commands are being modified from the network, then the interface card invokes a hierarchy to determine which reference is to be passed to the inverter as its main reference command.

The S## function code hierarchy is listed from highest to lowest priority in Table 12.

Table 12: S## Function Code Hierarchy

S## Function Code	Hierarchy Priority	Description
S01	Highest	frequency reference / per-unit
S05	2nd Highest	frequency reference / Hz
S19	3rd Highest	speed command
S02	4th Highest	torque command
S03	5th Highest	torque current command
S13	Lowest	PID command

The highest-priority S## function code with a non-zero value will be used as the inverter's main reference command.

3.2 Inverter Reaction to Network Timeout Conditions

Function codes o27 and o28 specify the inverter's reaction when a network timeout occurs. Table 13 lists the settings for o27 and o28.

Table 13: Inverter Reaction to Network Timeout Conditions (Function Codes o27 and o28)

o27 Value	o28 Value	Inverter reaction when a timeout occurs	Remarks
0, 4 to 9	---	Immediately coast to a stop and trip ErS.	
1	0.0s to 60.0s	After the time specified by o28, coast to a stop and trip ErS.	
2	0.0s to 60.0s	If the communications link is restored within the time specified by o28, ignore the communications error. After the timeout, coast to a stop and trip ErS.	
3, 13 to 15	---	Maintain present operation, ignoring the communications error (no ErS trip).	
10	---	Immediately decelerate to a stop. Trip ErS after stopping.	Inverter function code F08 specifies the deceleration time
11	0.0s to 60.0s	After the time specified by o28, decelerate to a stop. Trip ErS after stopping.	Same as above
12	0.0s to 60.0s	If the communications link is restored within the time specified by o28, ignore the communications error. After the timeout, decelerate to a stop and trip ErS.	Same as above

For details regarding the interface-specific timeout behavior and configuration, please refer to section 5.4.1.

4 FUNCTION CODE NUMBERING AND BEHAVIOR

4.1 Register Numbers

All accessible inverter function codes can be referenced by their Modbus register indices, as defined in the **RS-485 User's Manual (24A7-E-0082)**, section 3 (Table 3.2). These same register numbers are used when accessing function codes via certain Ethernet protocols. The terms "function code" and "register" refer to data stored on the inverter and will be used interchangeably throughout this documentation. The max supported register number is 13668. Because the RS-485 User's Manual contains information for several Fuji inverter families, the relevant information will be paraphrased here for the specific case of the FRENIC-Ace.

All inverter function codes are exposed as register indices according to a mathematical conversion formula which combines two elements (a function code group number and function code offset) to create a unique register number for each function code. Each function code group ("E" / Extension Terminal Functions, for example) is assigned a specific function code group number (refer to Table 14). Each function code also has an offset number, which is the function code without the leading letter (the offset number for function code E05, for example, is 5). To determine the register number for a given function code, therefore, the group number is first multiplied by 256, then added to the offset number plus 1. This operation is expressed mathematically via Equation 1.

$$register = (group\ number\ x\ 256) + offset\ number + 1 \qquad \textbf{Equation 1}$$

As an example, let's calculate the register number for output frequency (function code M09). According to Table 14, the group number for the "M" function code group is 8. It is also evident that the offset number for M09 is 9. Inserting the group number and offset number into Equation 1, we arrive at the result indicated in Equation 2.

$$(8\ x\ 256) + 9 + 1 = 2058 \qquad \textbf{Equation 2}$$

While manually calculating all of the register numbers for the function codes of interest is certainly possible by using Equation 1, it may be more convenient to simply reference the "Register" column on the monitor tab of the default web interface (refer to section 6.2.3).

Note that not all of the available registers that exist in the interface card's register map have corresponding function codes that exist in the inverter. In other words, if a read from or write to a register number that does not correspond to an existing inverter function code takes place, the read/write may be successful (depending on the specific register accessed; refer to section 4.2), but the data will have no meaning. This feature is beneficial in situations where the accessing of non-contiguous registers can be made more efficient by accessing an all-inclusive block of registers (some of which correspond to inverter function codes and some of which do not), while only manipulating those in your local programming that are known to exist.

Table 14: Function Code-to-Register Conversion Examples

Function Code Group		Group Number	Register Example Using Equation 1
Code	Name		
F	Fundamental Functions	0	F00: (0 x 256) + 0 + 1 = 1 ... F07 (acceleration time 1): (0 x 256) + 7 + 1 = 8 ... F99: (0 x 256) + 99 + 1 = 100
E	Extension Terminal Functions	1	E00: (1 x 256) + 0 + 1 = 257 … E98 (terminal [FWD] function): (1 x 256) + 98 + 1 = 355 E99: (1 x 256) + 99 + 1 = 356
C	Control Functions	2	C00: (2 x 256) + 0 + 1 = 513 … C20 (jogging frequency): (2 x 256) + 20 + 1 = 533 … C99: (2 x 256) + 99 + 1 = 612
P	Motor 1 Parameters	3	P00: (3 x 256) + 0 + 1 = 769 … P03 (motor 1 rated current): (3 x 256) + 3 + 1 = 772 … P99: (3 x 256) + 99 + 1 = 868
H	High Performance Functions	4	H00: (4 x 256) + 0 + 1 = 1025 … H11 (deceleration mode): (4 x 256) + 11 + 1 = 1036 … H99 (4 x 256) + 99 + 1 = 1124
A	Motor 2 Parameters	5	A00: (5 x 256) + 0 + 1 = 1281 … A05 (motor 2 torque boost): (5 x 256) + 5 + 1 = 1286 … A99: (5 x 256) + 99 + 1 = 1380
o	Operational Functions	6	o00: (6 x 256) + 0 + 1 = 1537 o01: (6 x 256) + 1 + 1 = 1538 … o99: (6 x 256) + 99 + 1 = 1636
S	Command Data	7	S00: (7 x 256) + 0 + 1 = 1793 … S05 (frequency command): (7 x 256) + 5 + 1 = 1798 … S99: (7 x 256) + 99 + 1 = 1892
M	Monitor Data 1	8	M00: (8 x 256) + 0 + 1 = 2049 … M09 (output frequency): (8 x 256) + 9 + 1 = 2058 … M99: (8 x 256) + 9 + 1 = 2148
r	Motor 4 Parameters	10	r00: (10 x 256) + 0 + 1 = 2561 … r02 (motor 2 base frequency): (10 x 256) + 6 + 1 = 2563 … r99: (10 x 256) + 99 + 1 = 2660

Function Code Group		Group Number	Register Example Using Equation 1
Code	Name		
U	Customizable Logic Functions	11	U00: (11 x 256) + 0 + 1 = 2817 … U99: (11 x 256) + 99 + 1 = 2916
J	Application Functions 1	13	J00: (13 x 256) + 0 + 1 = 3329 … J03 (PID proportional gain): (13 x 256) + 3 + 1 = 3332 … J99: (13 x 256) + 99 + 1 = 3428
y	Link Functions	14	y00: (14 x 256) + 0 + 1 = 3585 … y98 (bus link function): (14 x 256) + 98 + 1 = 3683 y99: (14 x 256) + 99 + 1 = 3684
W	Monitor Data 2	15	W00: (15 x 256) + 0 + 1 = 3841 … W32 (PID output): (15 x 256) + 32 + 1 = 3873 … W99 (15 x 256) + 99 + 1 = 3940
X	Alarm Data 1	16	X00 (alarm history / latest): (16 x 256) + 0 + 1 = 4097 … X99: (16 x 256) + 99 + 1 = 4196
Z	Alarm Data 2	17	Z00: (17 x 256) + 0 + 1 = 4353 … Z53 (3rd last alarm torque): (17 x 256) + 53 + 1 = 4406 … Z99: (17 x 256) + 99 + 1 = 4452
b	Motor 3 Parameters	18	b00: (18 x 256) + 0 + 1 = 4609 … b12 (motor 3 starting frequency): (18 x 256) + 12 + 1 = 4621 … b99: (18 x 256) + 99 + 1 = 4708
d	Application Functions 2	19	d00: (19 x 256) + 0 + 1 = 4865 … d24 (zero speed control): (19 x 256) + 24 + 1 = 4889 … d99: (19 x 256) + 99 + 1 = 4964
W1	Monitor Data 3	22	W100: (22 x 256) + 0 + 1 = 5633 … W199 (22 x 256) + 99 + 1 = 5732
W2	Monitor Data 4	23	W200: (23 x 256) + 0 + 1 = 5889 … W299 (23 x 256) + 99 + 1 = 5988
o1	Operational Functions	37	o100: (37 x 256) + 0 + 1 = 9473 … o199: (37 x 256) + 99 + 1 = 9572
U1	Customizable Logic Functions	39	U100: (39 x 256) + 0 + 1 = 9985 … U199: (39 x 256) + 99 + 1 = 10084

Function Code Group		Group Number	Register Example Using Equation 1
Code	Name		
J1	PID Control 1	48	J100: (48 x 256) + 0 + 1 = 12289 … J199: (48 x 256) + 99 + 1 = 12388

4.2 Scanned Function Codes

The interface card provides network access to the specified list of function codes contained in the *param.xml* file located in the "*WEB*" folder of the interface card's file system. These function codes are constantly being read and/or written (as applicable), and their current values are therefore mirrored in the interface card's internal memory. Only those function codes specified in the *param.xml* file will represent meaningful values.

The principle disadvantage of scanned function codes is that write data checking is not available. This means that when the value of a scanned function code is modified via a network protocol or via the web browser's monitor tab, the interface card itself is not able to determine if the new value will be accepted by the inverter (the value may be out-of-range, or the inverter may be in a state in which it will not accept new values being written via communications, etc.) For example, if a write is performed to a scanned command function code with a data value that is out-of-range, the interface card will not generate a corresponding error. However, if end-to-end confirmation of such data writes is required, then the function code can be read over the network at a later time to confirm that the written value "took hold" in the inverter.

Accesses to any function code (?00…?99, where "?" is any valid function code group letter from Table 14) will always be successful. Even if an inverter function code corresponding to a given register does not exist in the *param.xml* file, the interface card still maintains a placeholder location in its internal mirroring memory for that function code. This feature allows for the block access of non-contiguous registers (function codes) as described in section 4.1. Care must be taken to utilize only the function codes that are known to exist and that are also specified in the *param.xml* file.

4.3 Commonly Used Function Codes

For a complete listing of all available function codes, their bit mappings, scaling values, etc., please refer to the **Fuji FRENIC-Ace Instruction Manual (INR-SI47-1733a-E)** and the **Fuji RS-485 User's Manual (24A7-E-0082)**. As a user convenience, the structures of the commonly-used "Operation command" (function code S06), "Operation status" (function code M14) and "Rotation Speed" (function code W08) are replicated here (refer to Table 15, Table 16 and Table 17, respectively).

Table 15: Structure of "Operation command" (Function code S06)

Data format [14] Operation command

15	14	13	12	11*1	10	9	8	7	6	5	4	3	2	1	0
RST	XR (REV)	XF (FWD)	0	EN	X9	X8	X7	X6	X5	X4	X3	X2	X1	REV	FWD
↑	General-purpose input		Unused	EN terminal	General-purpose input									FWD: Forward command	
Alarm reset														REV: Reverse command	

*1 bit11: The EN terminal is a bit dedicated for monitor and the terminal command cannot be input through communications. (Applicable only with FRN☐☐G1☐-☐E and FRN☐☐G1☐-☐A.)

(All bits are turned ON when set to 1.)

(Example) When S06 (operation command) = FWD, X1 = ON

$0000\ 0000\ 0000\ 0101_b = 0005_H$ Consequently, ⇒

00_H	05_H

Table 16: Structure of "Operation status" (Function code M14)

Data format [16] Operation status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	0	0	RL	ALM	DEC	ACC	IL	VL	0	NUV	BRK	INT	EXT	REV	FWD

(All bits are turned ON or become active when set to 1.)

Bit	Symbol	Description	Support*1				Bit	Symbol	Description	Support*1			
			Mini	Eco	Multi	MEGA				Mini	Eco	Multi	MEGA
0	FWD	During forward rotation	O	O	O	O	8	IL	During current limiting	O	O	O	O
1	REV	During reverse rotation	O	O	O	O	9	ACC	During acceleration	O	O	O	O
2	EXT	During DC braking (or during pre-exciting)	O	O	O	O	10	DEC	During deceleration	O	O	O	O
3	INT	Inverter shut down	O	O	O	O	11	ALM	Alarm relay (for any fault)	O	O	O	O
4	BRK	During braking (fixed to 0 for FRENIC-Mini)	×	O	O	O	12	RL	Communicati ons effective	O	O	O	O
5	NUV	DC link circuit voltage established (0 = undervoltage)	O	O	O	O	13	0	–	×	×	×	×
6	TL	During torque limiting	×	×	O	O	14	0	–	×	×	×	×
7	VL	During voltage limiting	O	O	O	O	15	BUSY	During function code data writing	O	O	O	O

*1 The "Support" column indicates whether each inverter type supports the corresponding bit or not. The symbol "O" means the code is supported and the symbol "X" means that the code is not supported (fixed to 0).

Table 17: Structure of "Rotation Speed" (Function code W08)

Data format [37] Floating point data (load rotation speed, etc.)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Exponent		Mantissa													

Exponent: 0-3 Mantissa: 1 to 9999

The value expressed by this format = the mantissa × $10^{(\text{exponent-2})}$

Numeric value	Mantissa	Exponent	$10^{(\text{exponent-2})}$
0.01 to 99.99	1 to 9999	0	0.01
100.0 to 999.9	1000 to 9999	1	0.1
1000 to 9999	1000 to 9999	2	1
10000 to 99990	1000 to 9999	3	10

5 FUJI CONFIGURATION STUDIO

5.1 Overview

The interface card is discovered, configured and updated by the Fuji Configuration Studio PC application (refer to Figure 8). The studio must be installed prior to connecting an interface card to the computer in order to ensure that the appropriate USB drivers are installed. The studio requires a USB connection for reading/writing a configuration and updating the firmware. Depending on the currently-active drivers, remote discovery, network setting, configuration, and firmware updates are also possible via Ethernet. To obtain the latest release of the Configuration Studio, please contact technical support. The remainder of this section will provide only a brief introduction to the configuration concepts. For protocol specific configuration, refer to the relevant protocol section.

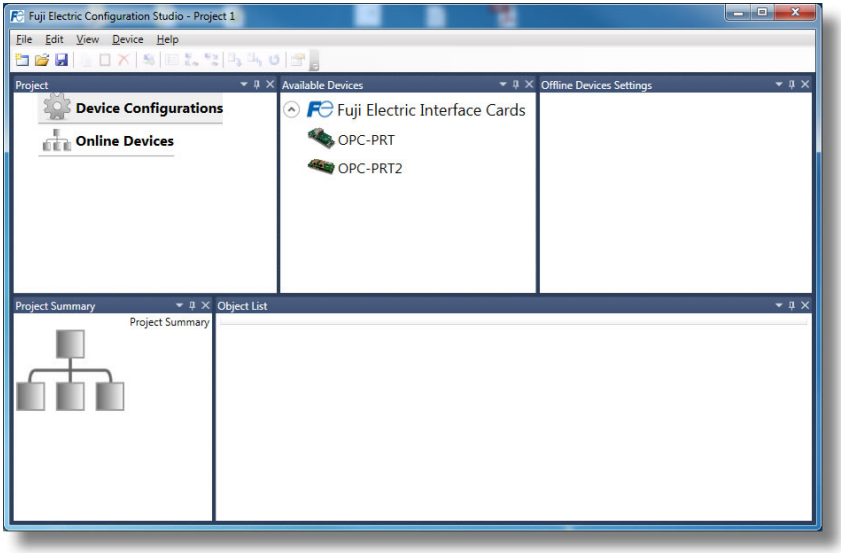


Figure 8: Fuji Configuration Studio

Creating a Device Configuration

A device can be added to the **Project** panel for configuration by first selecting the **Device Configurations** list heading and then:

- Double-clicking on the device in the **Available Devices** panel.
- Right-clicking on the device in the **Available Devices** panel and choosing **Add** from the context-sensitive menu.
- Hitting the <ENTER> key on the keyboard when the device is selected in the **Available Devices** panel.
- Dragging the device from the **Available Devices** panel into the **Project** panel.
- Selecting it and selecting **Add Selected Device** from the **Edit** menu.
- Selecting it and clicking the **Add** button in the toolbar.

The device will then be added to the list of **Device Configurations**.

Going Online with a Device

All connected devices are automatically added to the **Discovered Devices** panel. This panel is shown by selecting the **Online Devices** list heading in the **Project** panel. To go online with a device:

- Double-click on it in the **Discovered Devices** panel.
- Right-click on it in the **Discovered Devices** panel and choose **Go Online** from the context-sensitive menu.

- Hit the <ENTER> key on the keyboard when the device is selected in the **Discovered Devices** panel.
- Drag it from the **Discovered Devices** panel into the **Project** panel.
- Select it and select **Go Online with Device** from the **Edit** menu.
- Select it and click the **Go Online** button in the toolbar.

When the studio goes online with a device, its configuration is automatically read. While the studio is online with a device, it will appear in green text in the **Discovered Devices** panel. The studio may be online with multiple devices simultaneously.

Uploading a Device's Configuration into a Project

The current configuration of an online device can be uploaded into the **Project** panel by selecting a device under the **Online Devices** list heading and then:

- Right-clicking on it and choosing **Upload Configuration** from the context-sensitive menu.
- Dragging it from the **Online Devices** heading into the **Device Configurations** heading.
- Selecting it and selecting **Upload Configuration to Project** from the **Device** menu.
- Selecting it and clicking the **Upload Configuration** button in the toolbar.

The device's configuration will then be added to the list of **Device Configurations**. Once the configuration is uploaded into the project, it may be modified.

Removing a Device Configuration from a Project

A configuration can be removed from a project by:

- Selecting the device in the **Project** panel and dragging it. A trash can icon will appear at the bottom of the **Project** panel, and dragging and dropping the device in the trash will remove it from the project.
- Hitting the <DELETE> key on the keyboard when the device is selected in the **Project** panel.
- Right-clicking on the device in the **Project** panel and choosing **Remove** from the context-sensitive menu.
- Selecting **Remove Selected Item** from the **Edit** menu when the device is selected.
- Clicking on the **Remove** button in the toolbar when the device is selected.

Going Offline with a Device

To go offline with a device:

- Select the device in the **Project** panel and drag it. A trash can icon will appear at the bottom of the **Project** panel, and dragging and dropping the device in the trash will go offline with it.
- Hit the <DELETE> key on the keyboard when the device is selected in the **Project** panel.
- Right-click on the device in the **Project** panel and choose **Go Offline** from the context-sensitive menu.
- Select **Go Offline with Device** from the **Edit** menu when the device is selected.
- Click on the **Go Offline** button in the toolbar when the device is selected.

Downloading a Configuration to a Device

To download a configuration to an online device, first select the device under the **Device Configurations** heading in the **Project** panel, and then navigate to **Device…Download Configuration to Device**. If the studio is currently online with only one compatible device, then the configuration will be downloaded to the online device. Otherwise, a device selection prompt is displayed to select which device to download the configuration to. Do not power off the device or interrupt the connection once the download is in progress as this may corrupt the firmware and/or the configuration.

Stop all other communication to the device when downloading.

Updating Firmware

The studio automatically manages firmware updates when going online with a device and downloading a configuration to a device. Download the latest studio to obtain the latest firmware. Do not power off the

device or interrupt the connection once the update is in progress as this may corrupt the firmware and/or the configuration.

Resetting an Online Device

To reset an online device, first select the device in the **Project** panel and then navigate to **Device…Reset Device**.

General Configuration Process

To configure a device, add the desired protocol(s) and configure any objects associated with the respective protocol(s). Any changes will take effect once the configuration is downloaded to a device.

Note that numeric values can be entered not only in decimal but also in hexadecimal by including “0x” before the hexadecimal number.

5.2 General Object Editing Activities

The following editing activities apply for all types of configuration objects and project elements.

Adding an Object

To add an object, click on an item (protocol driver or Node, for example) in the **Project** panel. Any available objects for that item will be listed in the **Available Objects** panel (the panel title depends on the currently-selected item). An object can then be added to the item by:

- Double-clicking on it.
- Right-clicking on it and choosing **Add** from the context-sensitive menu.
- Hitting the <ENTER> key on the keyboard when the object is selected.
- Dragging it into the **Project** panel.
- Selecting it and selecting **Add Selected Device** from the **Edit** menu.
- Selecting it and clicking the **Add** button in the toolbar.

The object’s configurable fields can then be populated with valid values (where applicable).

Viewing an Object

In the **Project** panel, select a parent object to display a summary of all its child objects. For example, selecting a protocol driver will display the driver’s configuration in the **Summary** panel and list of current objects in the **Object List** panel.

Updating an Object

To update an object, select the object in the **Project** panel and make any required changes in the **Settings** panel.

Deleting an Object

An object can be deleted by performing one of the three following actions:

- Selecting the object in the **Project** panel and dragging it. A trash can icon will appear at the bottom of the **Project** panel, and dragging the object to the trash will then delete it from the project.
- Hitting the <DELETE> key on the keyboard when the object is selected in the **Project** panel.
- Right-clicking on the object in the **Project** panel and choosing **Remove** from the context-sensitive menu.
- Selecting **Remove Selected Item** from the **Edit** menu when the object is selected.
- Clicking on the **Remove** button in the toolbar when the object is selected.

Note that this action cannot be undone. Deleting an object will also delete all of its child objects.

Copying and Pasting an Object

To copy an object, first click on an item in the **Project** panel. An object can then be copied by:

- Right-clicking on it and choosing **Copy** from the context-sensitive menu.
- Pressing the <CTRL+C> keys on the keyboard.
- Holding the <CTRL> key and dragging the item to the desired location in the **Project** panel.

- Dragging the item to a new location under a different parent object in the **Project** panel.
- Selecting **Copy Selected Item** from the **Edit** menu.
- Clicking on the **Copy** button in the toolbar.

To paste an object, first click on an item at the desired location in the **Project** panel. An object can then be pasted by:

- Right-clicking on it and choosing **Paste** from the context-sensitive menu.
- Pressing the <CTRL+V> keys on the keyboard.
- Dropping an item onto the desired location in the **Project** panel after holding the <CTRL> key and dragging the item.
- Dropping an item onto a new location under a different parent object in the **Project** panel after dragging the item.
- Selecting **Paste Item** from the **Edit** menu.
- Clicking on the **Paste** button in the toolbar.

After pasting an object, the object's configurable fields can then be modified with valid values (where applicable).

Note that the studio allows you to copy and paste items between different locations, including different devices. This is useful for copying partial configurations from one device to another.

Reordering Objects

Objects can be reordered in the **Project** panel by dragging the item to the desired location. If the item is dragged outside of the items in the project tree, it will be moved to the end.

5.3 Ethernet Settings

The **Ethernet Settings** panel contains Ethernet-related items that are not specific to any given protocol. These settings must be appropriately configured regardless of any Ethernet control protocols that may be enabled. The **Ethernet Settings** panel is then available whenever the **Ethernet** port is selected in the **Project** panel.

5.3.1 Authentication

Be sure to make a note of the new settings whenever authentication credentials are changed, as they must be entered whenever the web page is accessed or an FTP session is initiated.

User Name

The username is case-sensitive and can contain letters ("a...z" and "A...Z") and numbers ("0...9").

Password

The password is case-sensitive and can contain letters ("a...z" and "A...Z") and numbers ("0…9").

5.3.2 Network Configuration

The card supports a static IP address. The IP Address, Subnet Mask and Default Gateway fields must be configured. Please consult with your network administrator for the proper settings of these fields.

5.4 Internal Logic Settings

5.4.1 Fail-safe Values

5.4.1.1 Overview

The card can be configured to perform a specific set of actions when network communications are lost (timeout event). This allows each inverter parameter to have its own unique "fail-safe" condition in the event of network interruption. Support for this feature varies depending on the protocol: refer to the protocol-specific section of this manual for further information.

There are two separate elements that comprise the timeout configuration:

- The timeout time

- Timeout Object configuration

5.4.1.2 Timeout Time

The timeout time is the maximum number of milliseconds for a break in network communications before a timeout will be triggered. This timeout setting is configured at the protocol level as part of a driver's configuration, and used by the protocol drivers themselves to determine abnormal loss-of-communications conditions. These conditions then trigger timeout processing events. If it is not desired to have a certain protocol trigger timeout processing events, then the protocol's timeout time may be set to 0 (the default value) to disable this feature.

For some protocols, the timeout time is set by the master device (PLC, scanner, etc.), and a timeout time setting is therefore not provided in the Configuration Studio's driver configuration. Additionally, not all protocols support timeout detection: refer to the protocol-specific sections of this manual for more information.

5.4.1.3 Timeout Object Configuration

A timeout object is used as part of the timeout processing to set certain parameters to "fail-safe" values. When a timeout event is triggered by a protocol, the timeout objects are parsed and written to the corresponding function code(s). The timeout object(s) will be executed sequentially from first to last. To add a timeout object, select the device in the **Project** panel, then add **Internal Logic…Fail-safe Values…Timeout Object**. The following paragraphs describe the configurable fields of a timeout object:

Description

This field is strictly for user reference: it is not used at any time by the device.

Function Code

Enter the function code.

Data Type

This is the size of valid values and is fixed to "16-Bit Unsigned" allows for a range of timeout values between 0 and 65535.

Value

Enter the "fail-safe" timeout value that the function code encompassed by this timeout object will be automatically written with upon processing a timeout event triggered by a protocol.

5.4.2 Fail-safe Example

This example will demonstrate how to add one timeout object which will assign a value of 2000 (20.00Hz) to function code S05 (frequency command). In the **Project** panel, select the device and add **Internal Logic…Fail-safe Values…Timeout Object** as shown in Figure 9. The red error indicators are normal at this stage as the **Timeout Object Settings** have not yet been configured.

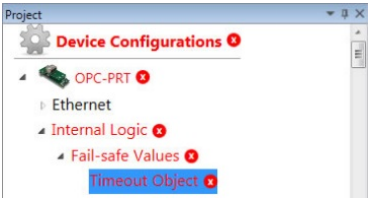


Figure 9: Timeout Object Project Panel

Next, configure the **Timeout Object Settings** as shown in Figure 10.

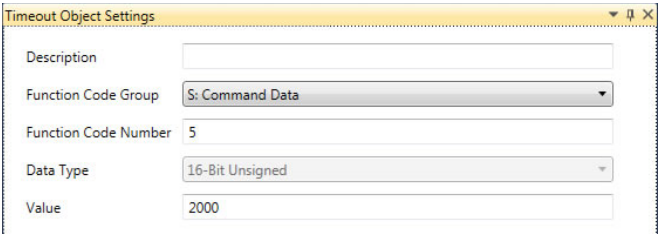


Figure 10: Timeout Object Settings

The example is complete.

5.5 Discovery over Ethernet

Depending on the currently-enabled driver, the Configuration Studio will automatically discover the device on the Ethernet network, regardless of whether or not the card's network settings are compatible with the subnet upon which they reside. All connected devices are automatically added to the **Discovered Devices** panel. This panel is shown by selecting the **Online Devices** list heading in the **Project** panel. In the **Discovered Devices** panel, discovered Ethernet devices will be listed under **Ethernet** and will display the firmware version in brackets and the current IP address in parentheses to the right of the device name (refer to Figure 11.)

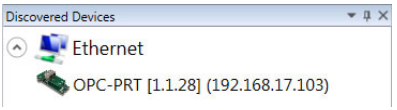


Figure 11: Configuration Studio Discovery over Ethernet

In order for the studio to discover devices, certain UDP Ethernet traffic (port 4334) must be allowed in and out of the computer, and firewall applications (such as Windows Firewall) are often configured to block such traffic by default. If the studio is unable to discover any devices on the current subnet, be sure to check the computer's firewall settings during troubleshooting, and add the studio as a program exception to the firewall configuration if necessary. It may be necessary to restart your PC before the new firewall configuration can take effect.

The network settings of a discovered card can be configured remotely by:

- Right-clicking on the device in the **Project** panel and choosing **Configure Network Settings…** from the context-sensitive menu.
- Selecting the device in the **Project** panel and navigating to **Device…Configure Network Settings...**

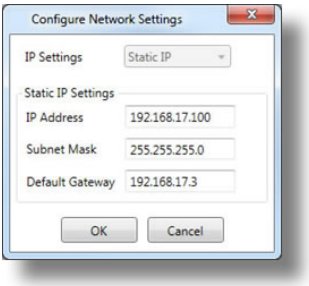


Figure 12: Remotely Configure Network Settings

The network settings pop-up should appear similar to Figure 12. Modify the network settings as necessary and click the OK button for the changes to take effect. Note that this will cause the device to become temporarily inaccessible and may trip the inverter.

5.6 Manage Device Parameters

The accessibility and scan priority of the inverter parameters can be adjusted (refer to Figure 13). This is an advanced feature and must only be used after consulting technical support to determine the appropriate settings for the target application. The **Manage Device Parameters** configuration window is found by:

- Right-clicking on the device in the **Project** panel and choosing **Manage Parameters…** from the context-sensitive menu.

- Selecting the device in the **Project** panel and navigating to **Device…Manage Device Parameters...**

A parameter is accessible and actively scanned (read from and written to the inverter) only if its corresponding checkbox is enabled. Likewise, a parameter is inaccessible if its checkbox is disabled.

Parameters that are accessed more frequently or require a faster update rate should be set to high priority. All other parameters should be set to low priority.

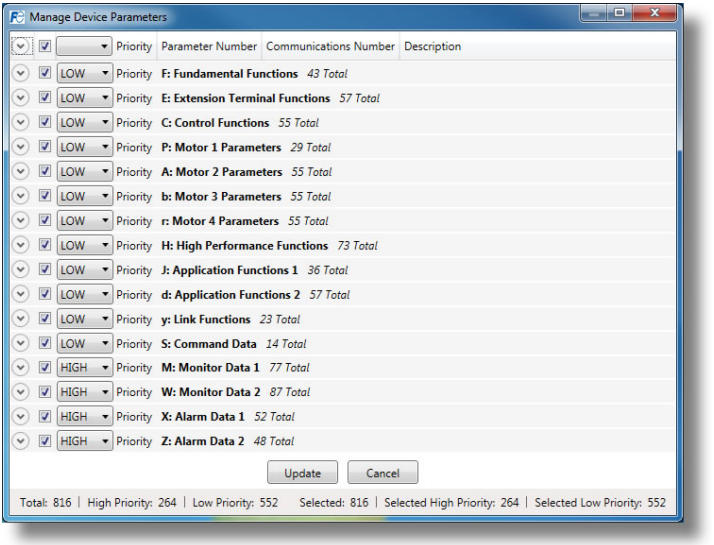

Manage Device Parameters
Priority
Parameter Number
Communications Number
Description
LOW Priority F: Fundamental Functions 43 Total
LOW Priority E: Extension Terminal Functions 57 Total
LOW Priority C: Control Functions 55 Total
LOW Priority P: Motor 1 Parameters 29 Total
LOW Priority A: Motor 2 Parameters 55 Total
LOW Priority b: Motor 3 Parameters 55 Total
LOW Priority r: Motor 4 Parameters 55 Total
LOW Priority H: High Performance Functions 73 Total
LOW Priority J: Application Functions 1 36 Total
LOW Priority d: Application Functions 2 57 Total
LOW Priority y: Link Functions 23 Total
LOW Priority S: Command Data 14 Total
HIGH Priority M: Monitor Data 1 77 Total
HIGH Priority W: Monitor Data 2 87 Total
HIGH Priority X: Alarm Data 1 52 Total
HIGH Priority Z: Alarm Data 2 48 Total
Update
Cancel
Total: 816 | High Priority: 264 | Low Priority: 552 Selected: 816 | Selected High Priority: 264 | Selected Low Priority: 552

Figure 13: Manage Device Parameters

5.7 Backup and Restore Parameters

The parameter settings can be backed up from the inverter and restored to the inverter (refer to Figure 14 and Figure 15). This allows for easy inverter cloning. The backup parameter list is stored as a CSV file. A parameter can be excluded from the list by disabling the corresponding checkbox. The parameter setting value can also be modified before the backup and restore is executed. The backup and restore parameter configurations are found by:

- Right-clicking on the device in the **Project** panel and choosing **Backup Parameters…** or **Restore Parameters…** from the context-sensitive menu.
- Selecting the device in the **Project** panel and navigating to **Device…Backup Parameters from Device...** or **Restore Parameters to Device…**

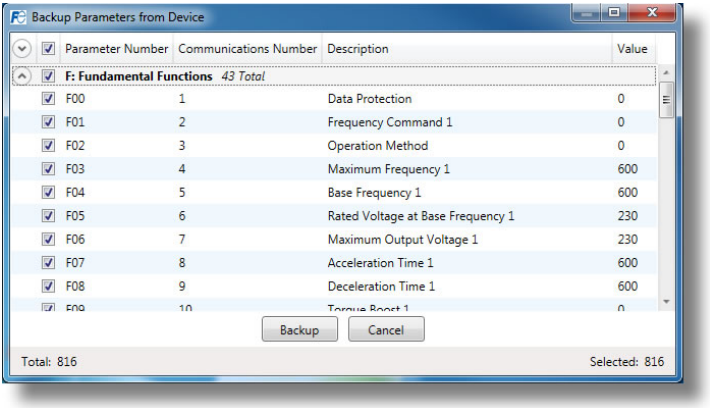

Backup Parameters from Device
Parameter Number
Communications Number
Description
Value
F: Fundamental Functions 43 Total
F00 1 Data Protection 0
F01 2 Frequency Command 1 0
F02 3 Operation Method 0
F03 4 Maximum Frequency 1 600
F04 5 Base Frequency 1 600
F05 6 Rated Voltage at Base Frequency 1 230
F06 7 Maximum Output Voltage 1 230
F07 8 Acceleration Time 1 600
F08 9 Deceleration Time 1 600
Backup
Cancel
Total: 816
Selected: 816

Figure 14: Backup Parameters

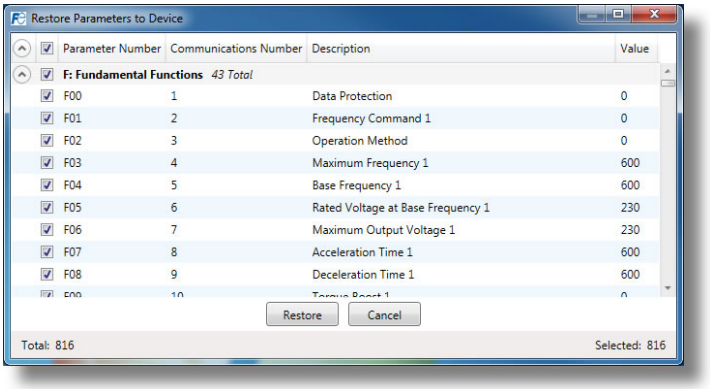

Restore Parameters to Device
Parameter Number
Communications Number
Description
Value
F: Fundamental Functions 43 Total
F00 1 Data Protection 0
F01 2 Frequency Command 1 0
F02 3 Operation Method 0
F03 4 Maximum Frequency 1 600
F04 5 Base Frequency 1 600
F05 6 Rated Voltage at Base Frequency 1 230
F06 7 Maximum Output Voltage 1 230
F07 8 Acceleration Time 1 600
F08 9 Deceleration Time 1 600
Restore
Cancel
Total: 816
Selected: 816

Figure 15: Restore Parameters

5.8 Restore Factory Settings

The interface card (connected via USB) can be restored to the factory settings. Note that the filesystem will be reformatted, which will destroy all custom modifications and configurations. Please backup the configuration before executing this feature. The factory settings can be restored by:

- Right-clicking on the device in the **Project** panel and choosing **Restore Factory Settings**.
- Selecting the device in the **Project** panel and navigating to **Restore Factory Settings**.

5.9 Help

Links to videos and documents can be found in the **Help** menu. Please review these links before contacting technical support for more in-depth assistance.

6 EMBEDDED WEB SERVER

6.1 Overview

The interface contains an embedded web server (also known as an HTTP server), which allows users to access the inverter's internal data in a graphical manner with web browsers such as Microsoft Internet Explorer or Mozilla Firefox. In this way, the inverter can be monitored and controlled from across the room or from across the globe. To access an interface's embedded web server, directly enter the target unit's IP address into the address (URL) field of your web browser. Refer to Figure 16 for a representative screenshot of the web server interface. In order to access the web server and view the parameter values, destination TCP ports 80 and 843 must be accessible from the client computer.

Note that in order to view the interface's web page, the free Adobe Flash Player browser plug-in is required. The plug-in can be downloaded from http://www.adobe.com.

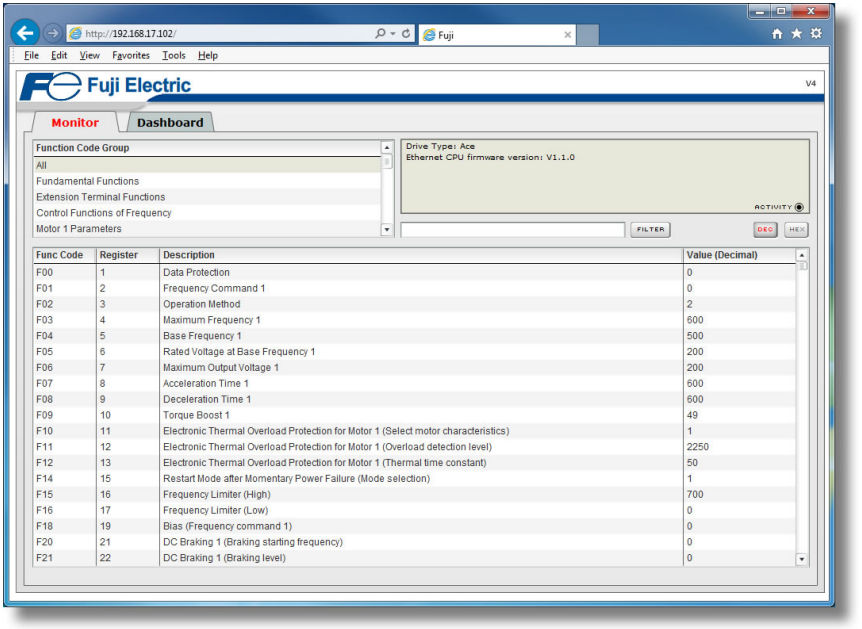

http://192.168.17.102/
Fuji
File Edit View Favorites Tools Help
Fuji Electric
V4
Monitor
Dashboard
Function Code Group
All
Fundamental Functions
Extension Terminal Functions
Control Functions of Frequency
Motor 1 Parameters
Drive Type: Ace
Ethernet CPU firmware version: V1.1.0
ACTIVITY
FILTER
DEC
HEX
Func Code | Register | Description | Value (Decimal)
F00 | 1 | Data Protection | 0
F01 | 2 | Frequency Command 1 | 0
F02 | 3 | Operation Method | 2
F03 | 4 | Maximum Frequency 1 | 600
F04 | 5 | Base Frequency 1 | 500
F05 | 6 | Rated Voltage at Base Frequency 1 | 200
F06 | 7 | Maximum Output Voltage 1 | 200
F07 | 8 | Acceleration Time 1 | 600
F08 | 9 | Deceleration Time 1 | 600
F09 | 10 | Torque Boost 1 | 49
F10 | 11 | Electronic Thermal Overload Protection for Motor 1 (Select motor characteristics) | 1
F11 | 12 | Electronic Thermal Overload Protection for Motor 1 (Overload detection level) | 2250
F12 | 13 | Electronic Thermal Overload Protection for Motor 1 (Thermal time constant) | 50
F14 | 15 | Restart Mode after Momentary Power Failure (Mode selection) | 1
F15 | 16 | Frequency Limiter (High) | 700
F16 | 17 | Frequency Limiter (Low) | 0
F18 | 19 | Bias (Frequency command 1) | 0
F20 | 21 | DC Braking 1 (Braking starting frequency) | 0
F21 | 22 | DC Braking 1 (Braking level) | 0

Figure 16: Embedded Web Server

6.2 Monitor Tab

6.2.1 Information Window

Figure 17 shows the Information Window that displays messages regarding the status of the interface card or web browser session. There is also an "ACTIVITY" indicator located in the lower-right hand corner of the Information Window that blinks periodically to show the status of data communication between the web browser and the interface card. If you do not observe the activity indicator blink at all for several seconds or more, it is possible that the web browser may have lost contact to the web server due to an inverter power cycle or a network problem. To reestablish communications, "refresh" your web browser.

Figure 17: Monitor Tab Information Window

6.2.2 Function Code Group Selection List

The Function Code Group Selection List is shown in Figure 18. Individual groups can be selected by clicking on the group name. Multiple groups may also be selected by holding down the CTRL key while clicking on the group names, or a range of groups can be selected by first selecting the starting group, and then holding down the SHIFT key while selecting the last group in the range. When a function code group is selected, the function codes contained in that group are displayed in the Function Code List (refer to section 6.2.3). The following function code groups are available:

Figure 18: Function Code Group Selection List

All: All function codes/registers are available.

Fundamental Functions: F function codes are available.

Extension Terminal Functions: E function codes are available.

Control Functions of Frequency: C function codes are available.

Motor 1 Parameters: P function codes are available.

Motor 2 Parameters: A function codes are available.

Motor 3 Parameters: b function codes are available.

Motor 4 Parameters: r function codes are available.

High Performance Functions: H, H1 function codes are available.

Application Functions 1: J function codes are available.

Application Functions 2: d function codes are available.

PID Control 1: J1 function codes are available.

Link Functions: y function codes are available.

Command Data: S function codes are available.

Monitor Data 1: M function codes are available.

Monitor Data 2: W function codes are available.

Monitor Data 3: W1 function codes are available.

Monitor Data 4: W2 function codes are available.

Alarm Data 1: X function codes are available.

Alarm Data 2: Z function codes are available.

Operational Functions: o, o1 function codes are available.

Customizable Logic Functions: U, U1 function codes are available.

6.2.3 *Function Code List*

The function code list is shown in Figure 19. The function codes that are displayed in the list at any given time depend on the function code groups that are currently selected (refer to section 6.2.2) and the filter (refer to section 6.2.4).

The first column of the Function Code List shows the inverter function code designation that is normally used when accessing a given function code via the inverter's keypad. Note that this column is for user convenience and inverter user's manual cross-reference.

The second column of the Function Code List shows the register number for the corresponding function code. Certain protocols require the use of a register number to access the function code (refer to section 4.1). The third column contains the function code descriptions, which are used by the filter function. The last column performs two functions: it displays the current value of the function code, and (for writable function codes) also allows changing the function code's value by clicking on the number in the value column and entering the new value.

Func Code	Register	Description	Value (Decimal)
F00	1	Data Protection	0
F01	2	Frequency Command 1	0
F02	3	Operation Method	0
F03	4	Maximum Frequency 1	600
F04	5	Base Frequency 1	600
F05	6	Rated Voltage at Base Frequency 1	230
F06	7	Maximum Output Voltage 1	230
F07	8	Acceleration Time 1	500
F08	9	Deceleration Time 1	500
F09	10	Torque Boost 1	0
F10	11	Electronic Thermal Overload Protection for Motor 1 (Select motor characteristics)	1
F11	12	Electronic Thermal Overload Protection for Motor 1 (Overload detection level)	300
F12	13	Electronic Thermal Overload Protection for Motor 1 (Thermal time constant)	50
F14	15	Restart Mode after Momentary Power Failure (Mode selection)	0
F15	16	Frequency Limiter (High)	700
F16	17	Frequency Limiter (Low)	0
F18	19	Bias (Frequency command 1)	0
F20	21	DC Braking 1 (Braking starting frequency)	0
F21	22	DC Braking 1 (Braking level)	0

Figure 19: Function Code List

Some items to keep in mind when interacting with the Function Code List are:

- When entering new function code values, be sure that the number being entered is appropriate for the currently-selected radix (refer to section 6.2.5).
- The column widths can be changed by dragging the vertical bars that separate the header row's cells.
- If you begin changing a function code value and then decide to abandon the change, pressing the ESC key on your keyboard will abandon the change and redisplay the current function code value.
- When editing a function code value, clicking someplace off the entry cell is equivalent to hitting the ENTER key.

6.2.4 *Function Code List Filter*

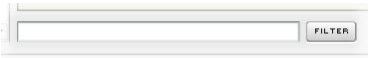


Figure 20: Function Code List Filter

A filter function provides Function Code List search capabilities. To use the filter function, simply type a word or portion of a word into the filter entry box and then click the "filter" button. Refer to Figure 20. The filter will then display only those function codes currently available in the Function Code List that satisfy the search criteria. Once a filter has been entered, it will continue to be applied to all information normally displayed in the Function Code List. To remove the filter, delete any characters contained in the filter entry box and then click the "filter" button.

6.2.5 Radix Selection

Figure 21 shows the radix selection buttons. These selection buttons allow changing the Function Code List “value” column data display and entry radix between decimal and hexadecimal formats.

Figure 21: Radix Selection

When “DEC” is selected, the “value” column heading will be “*Value (Decimal)*”, current function code values will be displayed in decimal, and values to be written to function codes must be entered in decimal format. For example, to change the inverter’s frequency command to 40.00Hz, enter the decimal value 4000.

Similarly, when “HEX” is selected, the “value” column heading will be “*Value (Hexadecimal)*”, current function code values will be displayed in hexadecimal, and values to be written to function codes must be entered in hexadecimal format. For example, to turn on bit #10 in the inverter’s operation command word, enter the hexadecimal number 0400.

6.3 Dashboard Tab

The Dashboard Tab provides access to a virtual keypad, as well as a variety of gauges, meters and graphs that can be configured to provide an at-a-glance graphical overview of critical application variables in real-time. A total of 10 gauge windows are available (two at a time), and each gauge window can be configured to display any scanned function code's value via one of six different gauge types. User-defined engineering units, scaling and range limits are also configurable. Refer to Figure 22.

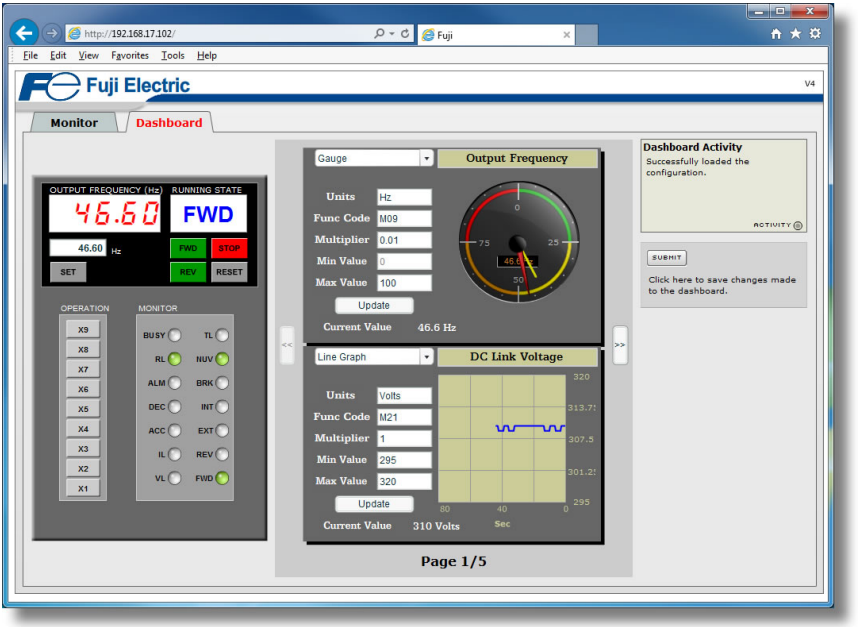


Figure 22: Dashboard Tab

6.3.1 Information Window

Figure 23 shows the Information Window, which displays various informational messages regarding the status of the Dashboard configuration parameters (loading or submitting).

Figure 23: Dashboard Tab Information Window

6.3.2 *Virtual Keypad*

A “virtual keypad” is displayed on the left-hand side of the dashboard tab, and acts as an interface for several useful pieces of control and monitor information. For an overview of the virtual keypad interface, refer to Figure 24. Note that it is recommended to suspend all external protocol-based communications with PLC’s, etc. when using the virtual keypad, as other protocols may simultaneously be writing to the inverter’s frequency command and operation command word, resulting in seemingly unpredictable behavior.

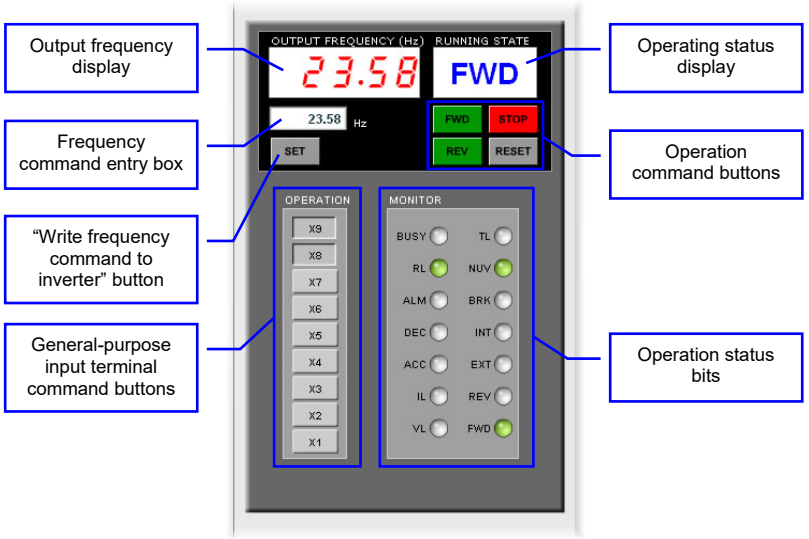


Figure 24: Virtual Keypad Overview

Output frequency display: Indicates the current output frequency of the inverter in large red numbers, as reported by inverter function code M09. The image in Figure 24 indicates that the associated inverter is currently running at 23.58Hz.

Frequency command entry box: Allows the user to enter a new frequency command for the inverter, which is subsequently scaled and written to inverter function code S05 when the “SET” button is clicked.

“SET” button: Clicking this button will scale and write the value contained in the frequency command entry box to inverter function code S05. Note that the inverter will use this frequency command as its master frequency reference only when configured accordingly (refer to section 3.1).

General-purpose input terminal command buttons: These buttons (labeled “X1” through “X9”) map to the corresponding bits in the inverter’s operation command word (function code S06, bit #2 ... bit #10). The usage of these bits varies depending on the configuration of inverter function codes E01 to E09. When a given button is clicked and shown in its depressed state, the corresponding bit is set to a “1”. When clicked again (and therefore shown in its non-depressed state), the corresponding bit is set to a “0”. As an example, the image in Figure 24 shows X1...X7 as OFF (“0”), and X8 and X9 as ON (“1”). Note that controlling these operation command word bits will only affect the inverter when it is configured accordingly (refer to section 3.1).

Operating status display: Indicates the current state of the inverter based on bits in the inverter operation status register (function code M14). Possible displays include “STOP”, “FWD”, “REV” and “FAULT”.

Operation command buttons: Clicking on these buttons enables different control actions to be imposed on the inverter, as follows:

- FWD: sets bit #0 (“FWD”) and clears bit #1 (“REV”) in the operation command word (function code S06).

- REV: sets bit #1 (“REV”) and clears bit #0 (“FWD”) in the operation command word (function code S06).
- STOP: clears both bit #0 (“FWD”) and bit #1 (“REV”) in the operation command word (function code S06).
- RESET: writes a value of “1” to function code S14 (alarm reset command). This will reset a faulted inverter regardless of the current operation command mode (H30, Y98 etc.) Note that if the inverter was running (the “FWD” or “REV” buttons were the last buttons pressed on the virtual keypad before the fault occurred), the STOP button must be clicked prior to clicking the RESET button in order to clear the FWD and REV bits in the operation command word. The inverter will ignore reset commands issued through function code S14 as long as a valid run command still exists in the operation command word.

Note that the inverter will follow the FWD, REV and STOP button commands only when configured accordingly (refer to section 3.1).

Operation status bits: These “virtual LEDs” map to the corresponding bits of the same name in the inverter’s operation status word (function code M14). When a given bit in the status word is “1”, then its corresponding indicator will be lit. The indicator will not be lit if its status word bit is “0”. As an example, the image in Figure 24 shows FWD (bit #0), NUV (bit #5) and RL (bit #12) ON, and all other bits OFF.

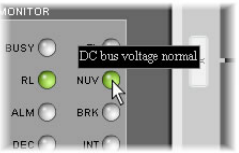


Figure 25: Virtual LED Tooltips

Tip Hovering the cursor over the virtual LEDs will bring up a tooltip which provides a brief summary of the indicated function. Refer to Figure 25.

6.3.3 Gauge Window Navigation

Figure 26 shows the two buttons that provide for navigation of the gauge windows. Gauge windows are displayed two at a time in the Dashboard Tab, and by clicking the “right” or “left” buttons, the gauge windows will scroll in the corresponding direction.

Figure 26: Gauge Window Navigation

6.3.4 Gauge Window Configuration

Each of the gauge windows can be independently configured to display a user-defined function code with a variety of flexible configuration options. While the behavior and presentation may vary slightly depending on the specific gauge chosen, all of the gauges share the following common elements (refer to Figure 27 for an example):

Gauge Selector: A drop-down selection box in the upper left-hand corner of the gauge window, which allows the user to select the type of gauge that will be displayed.

Title: A text entry box located above the gauge, in which the user can enter a descriptive gauge title comprised of up to 16 characters.

Units: A text entry box in which the user can enter an engineering units string comprised of up to 8 characters. This units string will be appended to all locations in the gauge window that display the designated function code’s current value.

Function Code: The designated function code whose value is to be reflected on the gauge. Note that only scanned function codes may be displayed in Dashboard gauges (refer to section 4.1 for a discussion of scanned function codes).

Multiplier: The multiplier value is a floating-point number that is used to scale the raw value of a function code. As its name suggests, the multiplier value is multiplied by the designated function code’s current raw value in order to calculate the gauge’s indicated value. Negative values can also be used if desired.

Min Value: The gauge’s minimum indicated value. Negative values can be used if desired (e.g. if a negative Multiplier attribute is used to generate a negative indicated value). Not all gauges allow adjustment of the min value.

Max Value: The gauge's maximum indicated value. Similar to the Min Value attribute, negative values can be used if desired. Indicated value characteristics can even be inverted by setting the Max Value attribute to a value less than the Min Value attribute.

Update Button: Clicking the update button will apply the current configuration attribute settings to the gauge. Note, however, that simply updating the gauge's current display properties does not write these settings to the interface card's file system. To save the current configuration of all the gauge windows to the file system, the Dashboard tab's "submit" button must be selected (refer to section 6.3.5).

Current Value: The current indicated value of the designated function code is numerically displayed with the configured Units string at the bottom of each gauge window.

The following is a summary of the different available gauge types:

Gauge: Refer to Figure 27. This type of meter implements a rotary dial-type display format. The indicated value and units are shown numerically on the face of the gauge, and via the red indicator needle. The yellow needle shows the previous indicated value, thereby providing a simple historical reference. The "Min Value" attribute is not configurable; this gauge always starts at 0.

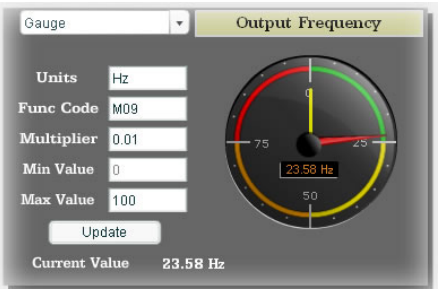


Figure 27: Gauge

BarGraph: Refer to Figure 28. This type of meter implements a linear bar graph display format. Hovering the mouse pointer over the red portion of the graph pops up a tooltip which displays the current indicated value and units.

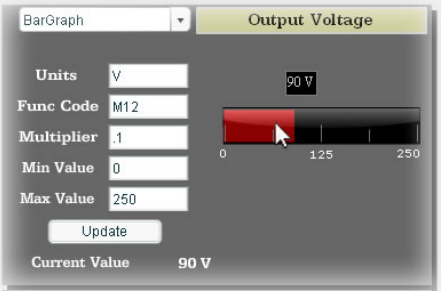


Figure 28: BarGraph

Meter: Refer to Figure 29. This type of meter implements a common panel meter-type display format. The units string is shown on the face of the meter. All raw function code values are interpreted as positive numbers (i.e. 0...0xFFFF equates to 0...65535_{10}.)

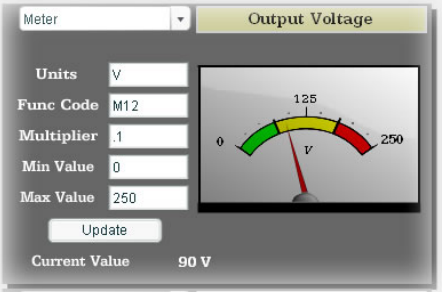


Figure 29: Meter

Pos/Neg Meter: Refer to Figure 30. Similar to the “meter” gauge, this type of meter also implements a common panel meter-type display format, but in this instance the indicated value can be positive or negative (two’s complement interpretation). In other words, raw function code values of 0...0x7FFF equate to 0...32767_{10}, and values of 0x8000...0xFFFF equate to -32768...-1. Because the meter placard is always centered around zero, the “Min Value” attribute is not configurable, and the “Max Value” attribute is used for both the maximum positive indicated value as well as the maximum negative indicated value.

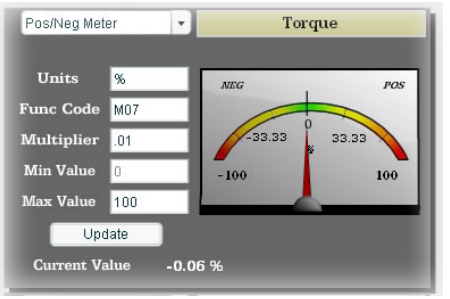


Figure 30: Pos/Neg Meter

Thermometer: Refer to Figure 31. This type of meter implements the universally-identifiable thermometer display format. Hovering the mouse pointer over the red “mercury” portion of the graph pops up a tooltip which displays the current indicated value and units.

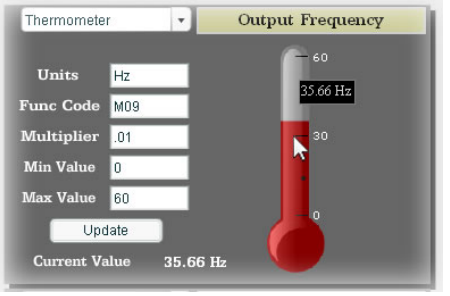


Figure 31: Thermometer

Line Graph: Refer to Figure 32. This type of graph implements a continuously-scrolling historical data logging line graph. Up to 80 seconds worth of historical data is available. Hovering the mouse pointer anywhere on the graph displays a vertical reference line at the corresponding time, and pops up a tooltip which displays the current indicated value at that time.

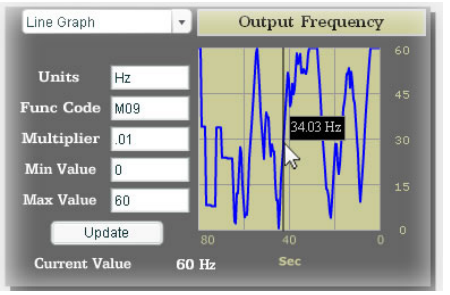


Figure 32: Line Graph

Tip

At times, it may be convenient to zoom in on a particular gauge or meter in order to more clearly see the indicator, or to fill the computer screen with a particular gauge's image. This can be easily accomplished with the web browser's Flash Player plug-in by right-clicking on the gauge and selecting the desired zoom level (refer to Figure 33).

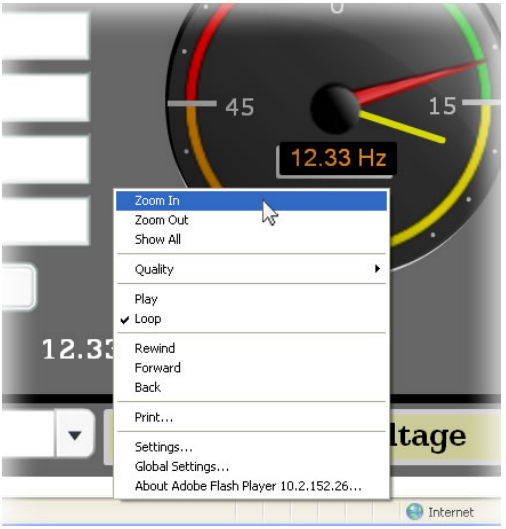


Figure 33: Zooming

6.3.5 Submitting Changes

Whenever any of the gauge window configuration items in the Dashboard Tab have been changed, the "submit" button located on the right-hand portion of the web page must be selected in order to write these settings to the interface card's file system. Refer to Figure 34. Note that submitting the Dashboard Tab configuration does not require rebooting of the interface card: the changes take effect immediately, and the interface card continues its operation without interruption.

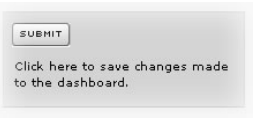


Figure 34: Submit Dashboard Changes

6.4 Customizing the Embedded Web Server

6.4.1 Customization Overview

It is possible for end-users to customize the embedded web server in order to create their own application-specific or corporate “look and feel”. Knowledge of authoring dynamic web content is required. Using windows explorer, it is possible to load customized web server content into the “WEB” folder on the interface card’s file system (refer to section 7.1.2). Usually, this web server content contains programming which implements the XML socket-based XTPro protocol (refer to section 6.4.2). Via XTPro, the embedded web server can gain access to any inverter parameter and the interface card file system resources, and manipulate them as required.

Notes

- There is an XML file located in the “WEB” folder called “*param.xml*”, which contains definitions for all inverter function codes that are available via the interface card. This file must not be removed, as it contains the definition of all available parameters not only for active web server content, but also for the interface card itself. All other files in the “WEB” folder may be deleted or replaced if desired by the user.
- The default HTML file targeted by the web server is “index.htm”. Therefore, when customizing the web server content, ensure that initial file “index.htm” exists.
- All files accessed by the web server itself must reside in the “WEB” folder. Note that this does not restrict active web server content to using only the “WEB” folder, however, as XTPro “read_file” and “write_file” commands can access any existing location on the file system.
- If the factory-default “WEB” folder contents need to be recovered (if they are accidentally deleted, for example), they can be downloaded from the device’s product page on the internet.
- Two simultaneous web server sessions are supported. Note that the number of available simultaneous web server sessions is independent of the number of available simultaneous XTPro XML sockets.

6.4.2 XTPro Overview

XTPro is an acronym for **X**ML **T**CP/IP **Pro**tocol. The XTPro specification is an application-layer (positioned at level 7 of the OSI model) messaging protocol that provides XML-based client/server communication via TCP port 843. Typically, XTPro is used for the implementation of graphical user interfaces (GUIs), such as advanced web servers or HMIs that have the ability to request information via XML sockets, and then manipulate and/or display the information in a rich application-specific manner.

XTPro is a request/response protocol that provides services specified by commands. For more information on XTPro, refer to the separate XTPro Specification. This section will cover the device-specific implementation of the XTPro protocol.

6.4.3 XTPro Web Browser-Based Implementation

A representative implementation based upon using a web browser as the client is detailed in Figure 35. In this scenario, the client application is developed by using an active web server authoring tool (such as Adobe Flash®). The active content is then embedded into one or more HTML files and loaded onto the device's file system (refer to section 6.4.1 for detailed information regarding customization of the web server content). Accessing the device's web server via a standard web browser then loads the active content, which initiates communication with the server.

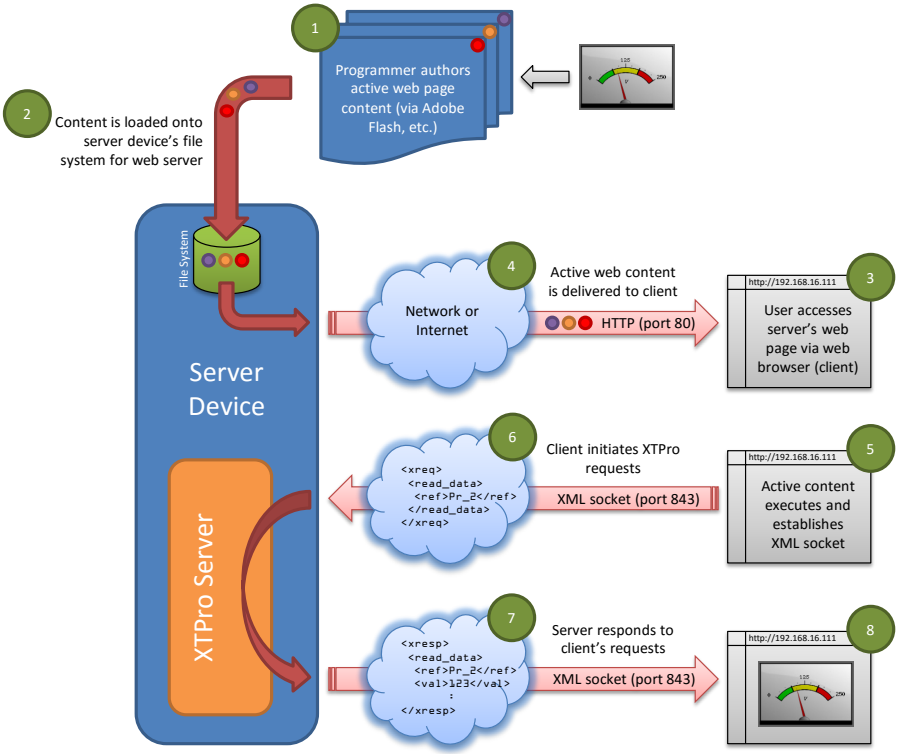


Figure 35: Web Browser-Based Implementation

6.4.4 XTPro HMI-Based Implementation

A representative implementation based upon a stand-alone HMI client is detailed in Figure 36. In this scenario, the client application is developed by using tools provided by the HMI manufacturer, and is hosted independently of the actual server device.

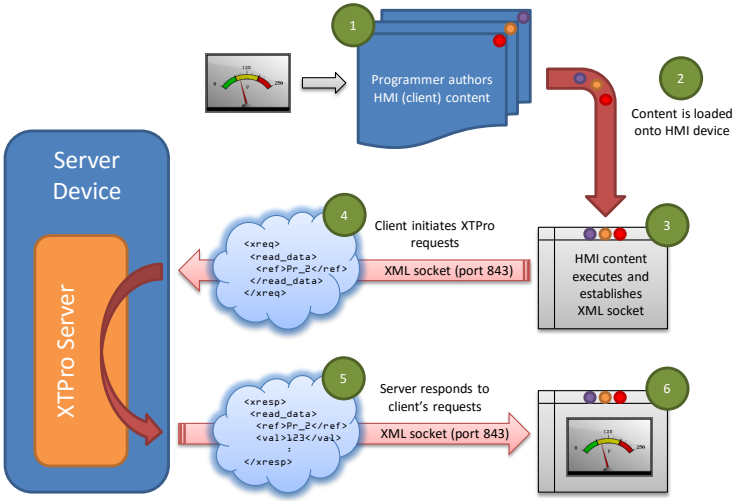


Figure 36: HMI-Based Implementation

6.4.5 XTPro Supported Commands

For a summary of XTPro commands, refer to Table 18.

Table 18: Supported XTPro Commands

Command	Supported	Notes
noop	Yes	-
vzn	Yes	Supports XTPro specification version 1
id	Yes	-
read_data	Yes	"reference" is the inverter's function code (e.g. "F07" for acceleration time #1), while "data_value" is a 16-bit hexadecimal value (e.g. "1F4" for a decimal value of 500)
write_data	Yes	
load_file	Yes	The absolute file path must start with a forward slash '/'
store_file	Yes	
reinit	No	Reinitializes only the configurable drivers and services (does not perform a complete device soft reboot)
auth	Yes	Authorization is not required
cov	Yes	COV notification messages are sent every 200ms

Notes

- Two simultaneous XTPro connections are available.

7 FILE SYSTEM & FIRMWARE

7.1 File System

7.1.1 Overview

The interface card's on-board file system is used by the application firmware. Currently, the application firmware's main use of the file system is to store XML-encoded configuration files and the embedded web server. The studio must be used to manage the configuration via USB or FTP. Do not manually access the configuration files unless instructed by technical support.

The configuration is only read at unit boot-up. Therefore, if a new configuration file is loaded, that unit must be rebooted for the new configuration take effect. Rebooting a unit can be performed by power-cycling the inverter in which the card is installed.

The embedded web server is customizable and is located in the "*WEB*" folder. All web page related items should reside in the "*WEB*" folder.

Interacting with the file system can be performed via USB (using a mini-B USB cable) as the interface card enumerates as a standard USB mass storage device ("flash drive"). The file system can also be accessed via FTP if the card has compatible network settings. Users can interact with the files on the interface card's file system in the same manner as though they were traditional files stored on a local or remote PC.

Note that the USB and FTP connection will prevent the file system from being accessed by other interfaces, such as the web server. Therefore, USB and FTP should only be connected when performing maintenance and configuration. USB and FTP should be disconnected while the card is running normally in a production environment.

7.1.2 USB with Windows Explorer

To use Microsoft Windows Explorer, first open either "Windows Explorer" or "My Computer". Refer to Figure 37. Note that the indicated procedure, prompts and capabilities outlined here can vary depending on such factors as the installed operating system and service packs.

The interface card will typically be displayed as a removable medium such as a "Removable Disk". Refer to Figure 38.

Figure 37: Accessing Windows Explorer

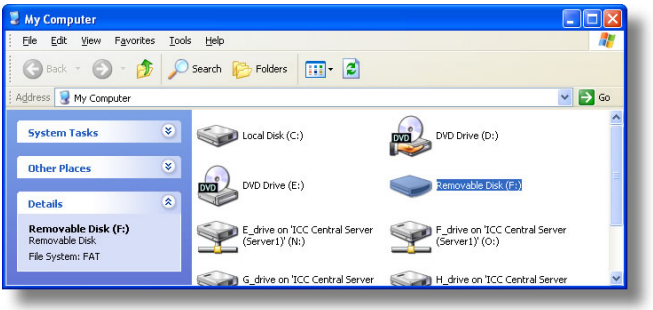


Figure 38: Removable Disk with Windows Explorer

Windows Explorer will then display the file system's contents (refer to Figure 39.) You can now perform normal file manipulation actions on the available files and folders (cut, copy, paste, open, rename, drag-and-drop transfers etc.) in the same manner as though you were manipulating any traditional file and folder stored on your computer's hard drive.

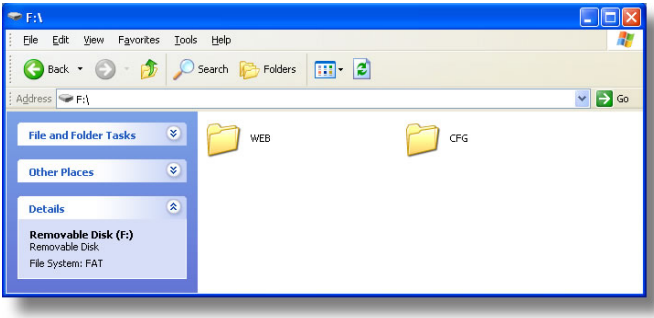


Figure 39: USB File Access via Windows Explorer

7.1.3 FTP with Windows Explorer

To use FTP with Microsoft Windows Explorer, first open either “Windows Explorer” or “My Computer”. Please note that the indicated procedure, prompts and capabilities outlined here can vary depending on such factors as the installed operating system, firewalls and service packs.

In the “Address” field, type in “ftp://admin:admin@” and then the IP address of the target interface card (if the user name and password have been changed from its default, then replace the first “admin” with the new user name and the second “admin” with the password.) Refer to Figure 40.

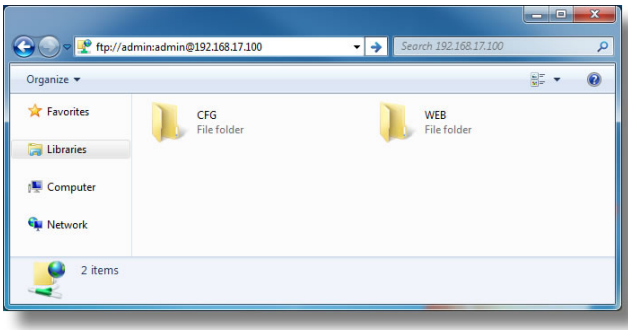


Figure 40: FTP via Windows Explorer

Note that the behavior of Windows Explorer FTP will vary from PC to PC. If you are having issues connecting FTP, there are other FTP client tools available such as Windows Command Prompt, Core FTP, FileZilla, SmartFTP etc. that can also be used to reliably access the card’s file system.

7.1.4 Loading New Web Server Content

The interface card’s web server resides in the file system and can be updated in the field (refer to section 6.4). This section will discuss how to update the default web server. The update procedure similarly applies to a custom web server. Web server updates may be released for a variety of reasons, such as improvements and added functionality. When using the default web server, it is always recommended to use the latest release.

Treat web server updates independently of firmware updates since web server updates may or may not be related to firmware updates. The latest default web server can be requested from technical support. It is suggested that users first check with technical support, and then periodically afterwards to determine if a new default web server has been released and is available to update their units.

Besides the new “WEB” folder containing the new web server, the update requires a USB connection as described earlier in this section. To update the web server, complete the following steps:

1. Navigate to the card's file system (see section 7.1.2 or 7.1.3).
2. Backup the "WEB" folder if desired by copying it to the local computer.
3. Delete the "WEB" folder from the card's file system.
4. Copy the new "WEB" folder to the card's file system.
5. Although it is not typical, if your *param.xml* file was specially modified (for a custom application, for example), it may be necessary to re-apply those modifications. Please consult technical support for any questions related to customized versions of *param.xml*.
6. Clear your internet browser's cache to ensure that the new web server content will be properly loaded from the interface card.

7.2 Firmware

7.2.1 Overview

The interface card's embedded firmware resides in flash memory that can be updated in the field. Firmware updates may be released for a variety of reasons, such as custom firmware implementations, firmware improvements and added functionality as a result of user requests. Additionally, it may be necessary to load different firmware onto the unit in order to support various protocols. In order to ensure that the firmware update is successful, and in the interest of equipment and personnel safety, it is strongly recommended to stop all of the card's production activities prior to initiating the firmware update procedure.

7.2.2 Update Procedure

1. Always back up your configuration to a PC for later recovery if necessary.
2. Download and install the latest Configuration Studio, which can be obtained from technical support.
3. Please be sure to read the firmware release notes and updated user's manual for any important notices, behavior precautions or configuration requirements prior to updating your firmware.
4. Ensure that the device is in a safe state prior to initiating the firmware update. The card may be temporarily inaccessible during the firmware update process.
5. *Locally via USB:* Connect a USB cable between the card and the PC and open the studio. If the studio contains newer firmware, it will automatically prompt you to update the firmware. Proceed with the firmware update.
6. *Remotely Via FTP:* Connect an Ethernet cable and ensure that the card has compatible network settings.
7. Once the firmware update process has started, do not interrupt the card as this may corrupt the firmware. Do NOT manually power-cycle the inverter or reboot the card. Do NOT disturb the USB or Ethernet (FTP) connection.
8. After the firmware update has been completed, the card will reset automatically. When the card boots up again, it will be running the new application firmware, which can be confirmed by observing the version displayed in **Device…Device Info** or the web server's information window (refer to section 6.2.1).
9. If new default web server content is available, load the new web server (refer to section 7.1.3).

8 PROTOCOL-SPECIFIC INFORMATION

This section will discuss topics that are specific to each of the supported protocols.

8.1 Modbus/TCP

8.1.1 Overview

The interface card supports Schneider Electric's Modbus/TCP protocol, release 1.0. The interface is conformance class 0 and partial class 1 and class 2 compliant, and allows up to 8 simultaneous Modbus/TCP client connections (sockets). Other notes of interest are:

- Supported Modbus/TCP functions are indicated in Table 19.

Table 19: Supported Modbus/TCP Functions

Function Code	Function	Modbus/TCP Class
1	Read coils	1
2	Read input status	1
3	Read multiple registers	0
4	Read input registers	1
5	Write coil	1
6	Write single register	1
15	Force multiple coils	2
16	Write multiple registers	0

- To calculate the register number for a function code, refer to section 4.1.
- Inverter registers can be addressed as holding registers (4X references) and input registers (3X references).
- Specific bits within inverter registers can be accessed as either coils (0X references) or discrete inputs (1X references).
- Because the transaction is handled locally within the interface card, write data checking is not available for scanned registers (refer to section 4.2.) For example, if a write is performed to a register with a data value that is out-of-range of the corresponding function code, no Modbus exception will be immediately returned.
- The "unit identifier" (UI) field of the request packets is ignored.
- Modbus/TCP should not be confused with Modbus (serial) over TCP. Modbus over TCP is not compatible with Modbus/TCP and is not supported.
- The driver can be configured to detect a timeout (communication loss) and perform a timeout action.

8.1.2 Holding & Input Registers

The inverter registers by default are mapped as both holding registers (4X) and input registers (3X) and are accessed by using the inverter register numbers described in section 4.1. The 4X and 3X only serve as a naming convention for holding register and input register respectively, and should NOT be included as part of the actual on-the-wire register number. To further clarify, Modbus register 42058 is the same as Modbus holding register 2058. The same description applies to input registers (3X).

For example, from a Modbus/TCP master's point of view, in order to access the output frequency (function code M09, register 2058) as a holding register, the Modbus/TCP master must execute the Read Multiple Registers function code and target register 2058. This will similarly apply when accessing an inverter function code as an Input Register.

8.1.3 Coil & Discrete Input Mappings

The Modbus/TCP driver provides read/write support for coils (0X references) and read-only support for discrete inputs (1X references). These will collectively be referred to from here on out as simply "discretes". Accessing discretes does not reference any new physical data: discretes are simply indexes

into various bits of existing registers. What this means is that when a discrete is accessed, that discrete is resolved by the interface into a specific register, and a specific bit within that register. The pattern of discrete-to-register/bit relationships can be described as follows:

Discrete 1...16 map to register #1, bit0...bit15 (bit0=LSB, bit15=MSB)
Discrete 17...32 map to register #2, bit0...bit15, and so on.

Arithmetically, the discrete-to-register/bit relationship can be described as follows: For any given discrete, the register in which that discrete resides can be determined by:

$$\text{register} = \left\lfloor \frac{\text{discrete} + 15}{16} \right\rfloor \qquad \textbf{Equation 3}$$

Where the bracket symbols "⌊ ⌋" indicate the "floor" function, which means that any fractional result (or "remainder") is to be discarded, with only the integer value being retained.

Also, for any given discrete, the targeted bit in the register in which that discrete resides can be determined by:

$$\text{bit} = (\text{discrete} - 1)\ \%\ 16 \qquad \textbf{Equation 4}$$

Where "discrete" ∈[1…65535], "bit" ∈[0…15], and "%" is the modulus operator, which means that any fractional result (or "remainder") is to be retained, with the integer value being discarded (i.e. it is the opposite of the "floor" function).

For clarity, let's use Equation 3 and Equation 4 in a calculation example. Say, for instance, that we are going to read coil #34. Using Equation 3, we can determine that coil #34 resides in register #3, as ⌊3.0625⌋ = ⌊3 r1⌋ = 3. Then, using Equation 4, we can determine that the bit within register #3 that coil #34 targets is (34-1)%16 = 1, as 33%16 = mod(2 r1) = 1. Therefore, reading coil #34 will return the value of register #3, bit #1.

8.1.4 Connection Timeout Options

In the studio's Project panel, navigate to **OPC-PRT…Ethernet…Modbus/TCP Server**. The following configuration options will determine the actions to be taken if the connection is abnormally terminated or lost. While this feature provides an additional level of fail-safe functionality for those applications that require it, there are several ramifications that must be understood prior to enabling this capability. Note that a certain degree of caution must be exercised when using the timeout feature to avoid "nuisance" timeouts from occurring.

Enable Supervisory Timer

This timer provides the ability for the driver to monitor timeout occurrences on the overall receive activity for all connections.

- The timer will start after receiving the first request. Once the timer is started, it cannot be disabled.
- If the driver experiences no receive activity for more than the **Timeout** time setting, then the driver assumes that the client or network has experienced some sort of unexpected problem, and will perform the **Timeout Action**.

Enable Connection Timer

This timer provides the ability for the driver to monitor timeout occurrences and errors within the scope of each client connection.

- If a particular open socket experiences no activity for more than the **Timeout** time setting, then the driver assumes that the client or network has experienced some sort of unexpected problem, and will close that socket and perform the **Timeout Action**.
- If a socket error occurs (regardless of whether the error was due to a communication lapse or abnormal socket error), the driver will perform the **Timeout Action**. Specifically, do not perform inadvisable behavior such as sending a request from the client device, and then closing the socket prior to successfully receiving the server's response. The reason for this is because the server will experience an error when attempting to respond via the now-closed socket. Always be sure to manage socket life cycles "gracefully", and do not abandon outstanding requests.

Timeout

Defines the maximum number of milliseconds for a break in network communications before a timeout event will be triggered.

Timeout Action

Select an action from the drop down menu:

"None"..... No effect. The inverter will continue to operate with the last available settings.

"Apply Fail-safe Values"..... Apply the fail-safe values as described in section 5.4.1.

"Fault Drive"..... The behavior will depend on the timeout conditions set by the inverter (function codes o27 and o28), which may result in an Er5 fault. Refer to section 3.2.

Enable Drive Fault Reset

This will clear the Er5 fault once communication is re-established. This option is only available if the **Timeout Action** is set to "Fault Drive".

8.1.5 Node Settings

There are no node settings. A node is simply a container for objects.

8.1.6 Holding/Input Register Remap Settings

In the studio's **Project** panel, add **OPC-PRT…Ethernet…Modbus/TCP Server…Node…Holding/Input Register Remap**.

The holding/input register remap objects are optional. By default, all inverter function codes are already mapped as both holding (4X) and input (3X) registers (refer to section 8.1.2). For user convenience, register remap objects can be created to map any inverter function code to holding/input register 5001 to 5050.

At times, it may be convenient to access inverter function codes in bulk Modbus transactions. This may be especially true in situations where it is desired to access certain function codes that are natively non-contiguous. For example, if it were desired to read the inverter's output frequency (function code M09, register 2058), operation status (function code M14, register 2063), and Life of cooling fan (function code M48, register 2097), this could be accomplished in two different ways:

1. Implement three separate Modbus read transactions, each one reading one register only, or
2. Implement one single Modbus read transaction, starting at register 2058 for a quantity of 40 registers. Then, pick out the registers of interest and ignore the rest of the response data.

While both of these methods will certainly work, neither one of them is optimized for the task at hand, which is to access three specific register values. A fully optimized solution can be realized by making use of the register remap objects. Non-contiguous inverter function codes can be grouped together in any order and accessed efficiently via the Modbus/TCP "read multiple registers" and "write multiple registers" function codes. The net effect is one of being able to transfer larger blocks of registers using fewer Modbus transactions, which results in improved network utilization and simpler data manipulation code on the Modbus master device.

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Remap Register

Remap register that maps to the specified inverter function code. Select from 5001 to 5050.

Function Code

Function code that is accessed by the **Remap Register**.

Data Type

Fixed to 16-Bit Unsigned. This is equivalent to two bytes.

8.2 EtherNet/IP

8.2.1 Overview

The EtherNet/IP protocol is an application-level protocol implemented on top of the Ethernet TCP/IP and UDP/IP layers. It shares its object model with ControlNet and DeviceNet through the Common Industrial Protocol (CIP). The card supports the EtherNet/IP server protocol, including the CSP server variant.

EtherNet/IP incorporates both the TCP and UDP layers of Ethernet in the transmission of data. EtherNet/IP uses TCP/IP only for explicit messaging; i.e. those messages in which the data field carries both protocol information and instructions for service performance. With explicit messaging, nodes must interpret each message, execute the requested task and generate responses. These types of messages can be used to transmit configuration, control and monitor data.

The UDP/IP protocol layer, which has the ability to multi-cast, is used for implicit (I/O) messaging. With I/O messaging, the data field contains only real-time I/O data; no protocol information is sent because the meaning of the data is pre-defined at the time the connection is established, which in turn minimizes the processing time of the node during run-time. I/O messages are short and have low overhead, which allows for the time-critical performance needed by controllers.

The interface card supports both explicit (UCMM and class 3) and implicit (class 1 I/O) messaging. When the EtherNet/IP server driver is added to the configuration, class 1 communication capability is enabled by default, and cannot be disabled. The driver continually listens for incoming class 1 connection requests. Further, two different types of I/O messaging are supported. One type (invoked when the client opens a connection to the interface using assembly instances 100 and 150) is entirely user-configurable. The other type (invoked when the client opens a connection to the interface using assembly instances 20 & 70 or 21 & 71) requires no user configuration and is included with the implementation of the AC/DC drive profile.

The following sections demonstrate specific examples of how to use EtherNet/IP to transfer data between the inverter and Allen-Bradley Logix-brand PLCs.

Some other notes of interest are:

- The interface card supports the EtherNet/IP protocol, as administered by the Open DeviceNet Vendor Association (ODVA).
- This product has been self-tested and found to comply with ODVA EtherNet/IP Conformance Test Software Version CT11.
- The interface card's product type code is 2 (AC Drive.)
- Supports unconnected messages (UCMM), and up to 16 simultaneous class 1 (I/O) or class 3 (explicit) connections.
- Class 1 implicit I/O supports both multicast and point-to-point (unicast) when producing data in the T→O direction.
- Point-to-point class 1 connected messages will be produced targeting the IP address of the device that instantiated the connection, UDP port 0x08AE (UDP port 2222).
- If a class 1 point-to-point connection is established in the (T→O) direction, no more class 1 connections can be established.
- If a class 1 connection's consuming half (O→T) times out, then the producing half (T→O) will also time-out and will stop producing.
- If a class 1 or class 3 connection timeout (communication loss) occurs, the driver can be configured to perform a timeout action. For class 1 connections, the timeout value is dictated by the scanner/client and is at least four times the RPI (Requested Packet Interval). For class 3 connections, the timeout value is also dictated by the scanner/client, but is typically a much larger value than for class 1 connections.

8.2.2 Server Settings

In the studio, navigate to **OPC-PRT…Ethernet…EtherNet/IP Server**.

Device Name

The device name is used for identification of a device on the EtherNet/IP network. This string is accessible as the "product name" attribute of the identity object. Enter a string between 1 and 32 characters in length.

8.2.3 Connection Timeout Options

In the studio's Project panel, navigate to **OPC-PRT…Ethernet…EtherNet/IP Server**. The following configuration options will determine the actions to be taken if the connection is abnormally terminated or lost. While this feature provides an additional level of fail-safe functionality for those applications that require it, there are several ramifications that must be understood prior to enabling this capability. Note that a certain degree of caution must be exercised when using the timeout feature to avoid "nuisance" timeouts from occurring.

Run/Idle Flag Behavior

EtherNet/IP clients (such as PLCs) have the option of adding a 32-bit "run/idle" header to all class 1 (I/O) data packets sent to devices. Bit 0 of this header is called the "run/idle flag" by the EtherNet/IP specification, and is intended to signify when the client is in a "running" state or an "idle" state. A running state (run/idle flag = Run) is indicated whenever the client is performing its normal processing (e.g. scanning its ladder logic). An idle state (run/idle flag = Idle) is indicated otherwise. For example, Allen Bradley ControlLogix PLCs will set their run/idle flag to Idle whenever their processor keyswitch is placed in the "PROG" position, presumably in preparation to receive a new application program from RSLogix 5000.

The behavior of EtherNet/IP devices when they receive I/O data from a controller with the run/idle flag set to Idle is not defined in the EtherNet/IP specification. The driver allows the option of two different behavioral responses when a run/idle flag = Idle condition is received, depending on the state of the *Invoke Timeout When Run/Idle Flag = Idle* checkbox.

- If the checkbox is cleared (default setting), then the driver will maintain the last I/O data values received from the client. For example, if a device mapped to the database was being commanded to run prior to the run/idle flag being set to Idle by the client, then it will continue to run.
- If the checkbox is checked, then the driver will perform the **Timeout Action**.

Timeout Action

Select an action from the drop down menu:

"None"..... No effect. The inverter will continue to operate with the last available settings.

"Apply Fail-safe Values"..... Apply the fail-safe values as described in section 5.4.1.

"Fault Drive"..... The behavior will depend on the timeout conditions set by the inverter (function codes o27 and o28), which may result in an Er5 fault. Refer to section 3.2.

Enable Drive Fault Reset

This will clear the Er5 fault once communication is re-established. This option is only available if the **Timeout Action** is set to "Fault Drive".

8.2.4 Generic Class 1 I/O Produced and Consumed Data Settings

The Produced Data Word and Consumed Data Word objects are only applicable when connecting to assembly instances 100 and 150 (generic I/O), which is typically the case. The Produced Data Word defines the structure of status data sent from the inverter back to the controller (T->O, target to originator). The Consumed Data Word objects will define the structure of the command data sent from the EtherNet/IP controller (for example, a ControlLogix PLC) to the inverter (O->T, originator to target). These objects allow the creation of custom-built I/O data. Up to 32 "command" function code values can be sent to the inverter, and up to 32 "status" function code values can be sent back to the controller. Therefore, up to 32 Produced and 32 Consumed Data Word objects can be created. If a consumed word offset is not defined, that data will be ignored by the inverter. If a produce word offset is not defined, the value will default to 0. The size of the actual I/O produced and consumed data is determined by the client upon initial connection establishment. Since a data word utilizes 2 bytes, the size must be an even number of bytes.

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Produced Data Word Offset

The value from the associated inverter function code will populate this word offset of the produced data that is to be sent to the client. It is recommend to start at word offset 0.

Consumed Data Word Offset

The consumed data received from the client at this word offset will contain the value to be written to the associated inverter function code. It is recommend to start at word offset 0.

Function Code

The inverter function code associated with the word offset. For the Produced Data Word object, enter a "status" function code to be monitored. For the Consumed Data Word object, enter a "command" function code that can be written.

Data Type

Each data word is fixed to 16-Bit Unsigned. This is equivalent to two bytes.

8.2.5 Generic Class 1 (I/O) Connection Access

Clients may access the class 1 endpoint by opening a connection to assembly instances 100 and 150. The structure of I/O consumed and produced data for this assembly instance pair is entirely user-configurable (refer to section 8.2.3). The generic class 1 I/O connection is mutually exclusive of the AC/DC drive profile class 1 I/O connection. For a generic class 1 I/O application example, refer to section 8.2.10.

8.2.6 AC/DC Drive Profile Class 1 (I/O) Connection Access

The interface card supports the ODVA AC/DC drive profile. No special EtherNet/IP configuration of the interface card is required when using the AC/DC drive profile: all that is needed is that the controller must target either assembly instances 20 & 70 or 21 & 71 in its connection parameters. The structure of I/O consumed and produced data for the AC/DC drive profile class 1 I/O is predefined and fixed (refer to Table 21 and Table 22) to 4 input bytes and 4 output bytes. It is highly recommended to complete the reading of this section to understand the data mapping and the implications of using the AC/DC drive profile. Note that when using the AC/DC drive profile class 1 I/O, the produced word and consumed word configuration do not apply (refer to section 8.2.3). For an AC/DC drive profile class 1 I/O application example, refer to section 8.2.12.

The AC/DC drive profile implementation provides support for several required CIP objects, which are specified in Table 20. While the various supported attributes of all of these objects are accessible via explicit messaging, the main intent of using the AC/DC drive profile is to interact with the predefined input and output assembly instances via an I/O connection. The structure of these assembly instances is defined by the EtherNet/IP specification in order to engender interoperability among different vendor's products. This section will focus primarily on the format of the AC/DC drive profile I/O assemblies supported by the interface card, and the inverter data which their various constituent elements map to.

Table 20: AC/DC Drive Profile-Related Objects

Class Code	Object Name
0x04	Assembly Object
0x28	Motor Data Object
0x29	Control Supervisor Object
0x2A	AC Drive Object

Table 21: Output Instances 20 and 21 Detail

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
20	0						Fault Reset		Run Fwd
	1								
	2	Speed Reference (Low Byte)							
	3	Speed Reference (High Byte)							
21	0		NetRef	NetCtrl			Fault Reset	Run Rev	Run Fwd
	1								
	2	Speed Reference (Low Byte)							
	3	Speed Reference (High Byte)							

Output Instance Mapping Detail

Run Fwd: forward rotation command (0=forward rotation off, 1=forward rotation on). Maps to inverter function code S06, bit 0 (function code S06 / operation command word, FWD bit).

Run Rev: reverse rotation command (0=reverse rotation off, 1=reverse rotation on). Maps to inverter function code S06, bit 1 (function code S06 / operation command word, REV bit).

Fault Reset: Inverter reset command (0=no action, 0→1 rising edge=reset). Maps to inverter function code S06, bit 15 (function code S06 / operation command word, RST bit).

NetCtrl: Not used (value is ignored).

NetRef: Not used (value is ignored).

Speed Reference: Inverter speed reference in RPM. Maps to function code S05 (frequency command). The speed reference component of the AC/DC drive profile output instances is always in units of RPM. Therefore, the interface card applies the RPM-to-Hz conversion indicated in Equation 5 in order to determine the appropriate frequency command value (in units of Hz) to be written to function code S05.

$$Hz = \frac{RPM \times number\ of\ motor\ poles}{120} \qquad \textbf{Equation 5}$$

The “number of motor poles” term which appears in the numerator of Equation 5 is obtained from the setting of inverter function code P01 (Motor number of poles). Note that the value of P01 is read by the interface card only at boot-up, so if the value of this function code is changed, then the interface card must be rebooted in order for it to read the new value from the inverter.

Table 22: Input Instances 70 and 71 Detail

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
70	0						Running1		Faulted
	1								
	2	Speed Actual (Low Byte)							
	3	Speed Actual (High Byte)							
71	0	At Reference	Ref From Net	Ctrl From Net	Ready	Running2 (REV)	Running1 (FWD)	Warning	Faulted
	1	Drive State							
	2	Speed Actual (Low Byte)							
	3	Speed Actual (High Byte)							

Input Instance Mapping Detail

Faulted: Inverter fault signal (0=not faulted, 1=faulted). Maps to function code M14, bit 11 (operation status word, ALM bit).

Warning: This bit is not used (it is always 0).

Running1 (FWD): Running forward status signal (0=not running forward, 1=running forward). Maps to function code M14, bit 0 (operation status word, FWD bit).

Running2 (REV): Running reverse status signal (0=not running reverse, 1=running reverse). Maps to function code M14, bit 1 (operation status word, REV bit).

Ready: Inverter ready signal (0=not ready, 1=ready). The Ready bit will be 1 whenever the Drive State attribute (see below) is in the Ready, Enabled or Stopping state.

CtrlFromNet: This bit is not used (it is always 0).

RefFromNet: This bit is not used (it is always 0).

AtReference: Up-to-speed signal (0=not up-to-speed, 1=up-to-speed). Set to 1 if the inverter is running (either Running1 = 1 or Running2 = 1) and both the ACC bit (bit #9) and DEC bit (bit #10) in the operation status word (function code M14) are 0.

Drive State: Indicates the current state of the Control Supervisor Object state machine. Refer to the ODVA EtherNet/IP specification (object library) for detailed information on the Control Supervisor Object state machine.

Speed Actual: Inverter operating speed in RPM. Maps to function code M09 (output frequency). The speed actual component of the AC/DC drive profile input instances is always in units of RPM. Therefore, the interface card applies the Hz-to-RPM conversion indicated in Equation 6 in order to determine the appropriate operating speed (in units of RPM) to be written to the network.

$$RPM = \frac{Hz \times 120}{number\ of\ motor\ poles} \qquad \textbf{Equation 6}$$

The “number of motor poles” term which appears in the denominator of Equation 6 is obtained from the setting of inverter function code P01 (Motor number of poles). Note that the value of P01 is read by the interface card only at boot-up, so if the value of this function code is changed, then the interface card must be rebooted in order for it to read the new value from the inverter.

8.2.7 Explicit Messaging Via Data Table Read/Write Services

Data table read (0x4C) and data table write (0x4D) services provide a direct method of accessing the inverter function codes by reference to “tag names”. Tags are read via the EtherNet/IP “data table read” service, and written via the EtherNet/IP “data table write” service.

To read data, the client must reference a starting “source element” and the “number of elements” to read. Similarly, to write data, the client must reference a starting “destination element” and the “number of elements” to write. The “number of elements” can be any quantity from 1 to the maximum allowable length, while the “source element” and “destination element” must be tag names constructed according to the naming conventions shown in section 8.2.8.

8.2.8 Inverter Function Code Access Tag Format

Any inverter function code can be accessed with its own unique tag name, or an array tag can be used to access a group of function codes with one PLC instruction. The "tag name" is essentially the ASCII representation of the function code itself. Tag names are generated according to the following structure:

[function code group][function code offset]

Where
[function code group] is a [1 to 2]-character field, and is the ASCII character(s) for the function code's group. The characters are case-sensitive. Refer to Table 14.

[function code offset] is a 2-character field corresponding to the function code offset. If the offset is less than 10, it must be pre-pended by 0. Valid offsets are "00" to "99".

Examples
"Acceleration time 1" F07
"Frequency command" S05
"Operation status" M14
"Output power" W22
"Life of cooling fan" W168

For explicit messaging examples, refer to sections 8.2.13, 8.2.14, 8.2.15, and 8.2.16.

8.2.9 ControlLogix Examples: Setup

This section will demonstrate how to initially setup a ControlLogix PLC (such as a 1756-L61) coupled with a 1756-ENBT/A communication interface (adjust this procedure according to your specific equipment). Later sections will provide specific read/write examples using this configuration with I/O or explicit messaging.

1) Run RSLogix 5000, and create a new configuration.

2) To add a 1756-ENBT/A to your I/O configuration, first switch to offline mode.

3) Right click on the I/O Configuration node in the controller organizer view and choose "New Module…"

4) The "Select Module" window will open.

5) Select the "1756-ENBT/A", and click "Create". Refer to Figure 41.

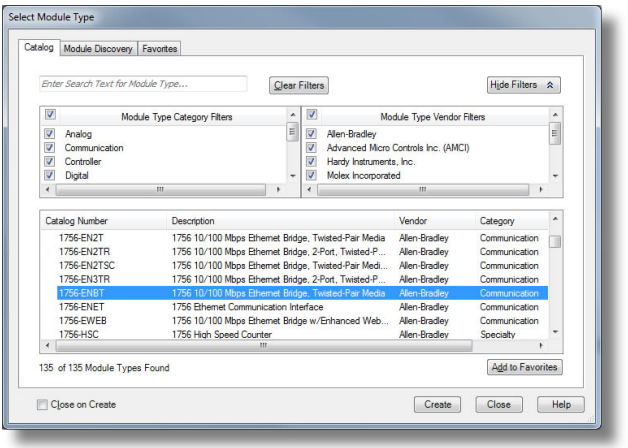


Figure 41: Adding a New Module

6) The "New Module" window will open. Refer to Figure 42.

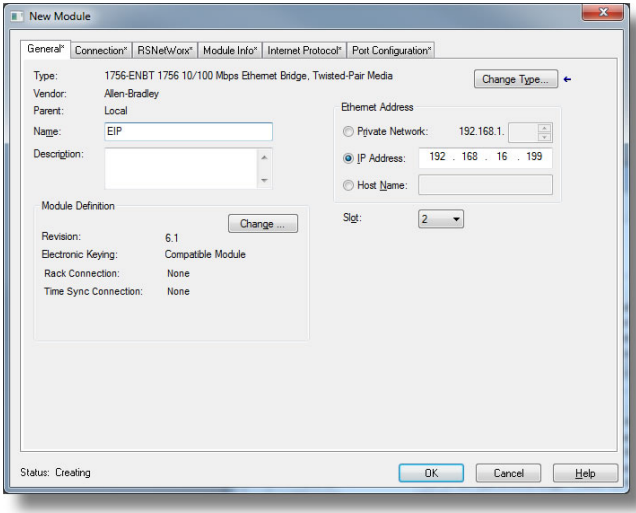


Figure 42: Identifying the New Module

7) Assign the Ethernet module a name (we will use “EIP”) and an IP address, deselect “Open Module Properties”, and click OK.

8) Download the configuration.

9) Switch to online mode. Right click on the 1756-ENBT/A module in the I/O Configuration and choose “Properties”.

10) Select the Internet Protocol tab from the Module Properties dialog box and confirm that the IP Settings are configured correctly.

8.2.10 ControlLogix Example: I/O Messaging

This section will demonstrate how to setup and use an EtherNet/IP I/O connection via vendor-specific assembly instances 100 & 150 or 20 & 70 or 20 & 71. EtherNet/IP I/O messaging allows the inverter’s function codes to be directly mapped into tags in the ControlLogix PLC. Once an I/O connection is established, it is automatically synchronized at an interval defined by the Requested Packet Interval (RPI).

1) Switch to offline mode.

2) Right click on the 1756-ENBT/A node under the I/O Configuration in the controller organizer view and choose “New Module…”

3) Choose “Generic Ethernet Module” in the Select Module dialog box and click “Create”. Refer to Figure 43.

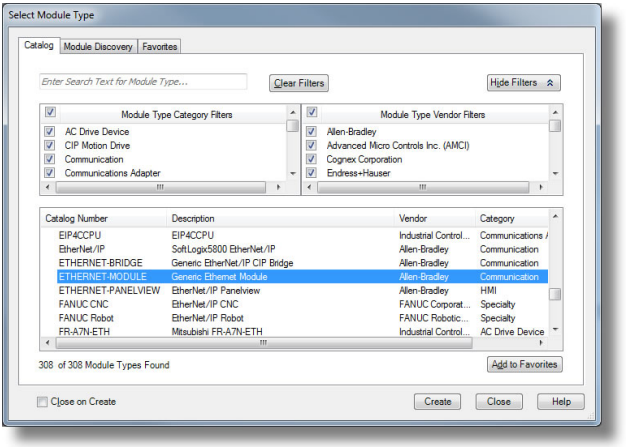


Figure 43: Adding a New Generic Ethernet Module

4) The module properties dialog box will open (refer to Figure 44). Enter a Name which will allow easy identification of the inverter on the network (the tags created in RSLogix 5000 will be derived from this Name). Because all inverter data is stored as 16-bit function codes, change the “Comm Format” selection to “Data-INT”. Enter the IP address of the targeted interface card.

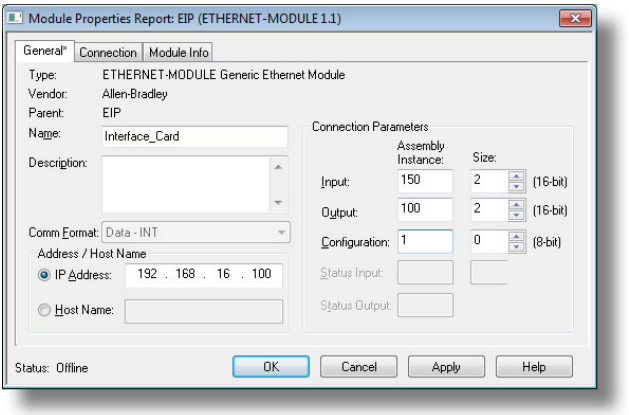


Figure 44: Interface Card Module Properties

In the “Connection Parameters” portion of the dialog box, enter the following information:

Input: The Input Assembly is the collection of monitor data that is produced by the interface card and is received as an input to the PLC. Its structure is defined by the Produced Data Configuration as described in section 8.2.3. The Input Assembly Instance must be set to 150 when connecting to the generic I/O assembly instances (or 70/71 when using the ODVA AC/DC drive profile), and the size must be set to the number of 16-bit function codes that we wish to receive from the interface card. For the purposes of this example, we are assuming that the default produced data word configuration, with two relevant function codes (M14 and M09). We therefore set the Input Size to 2.

Output: The Output Assembly is the collection of command & configuration data that is sent as an output from the PLC and consumed by the interface card. Its structure is defined by the Consumed Data Configuration as described in section 8.2.3. The Output Assembly Instance must be set to 100 when connecting to the generic I/O assembly instances (or 20/21 when using the ODVA AC/DC drive profile), and the size must be set to the number of 16-bit function codes that we wish to send to the interface card. For the purposes of this example, we are assuming that the default

consumed data word configuration, with two relevant function codes (S06 and S05). We therefore set the Output Size to 2.

Configuration: The Configuration Assembly Instance is unused, and its instance number and size are therefore irrelevant (you can just enter "1" and "0", respectively).

When done, click "OK".

5) You should now see the new module (named "ETHERNET-MODULE Interface_Card") in the 1756-ENBT/A branch under the I/O Configuration in the controller organizer view. Right click on this new module, choose "Properties", and select the Connection tab. Refer to Figure 45.

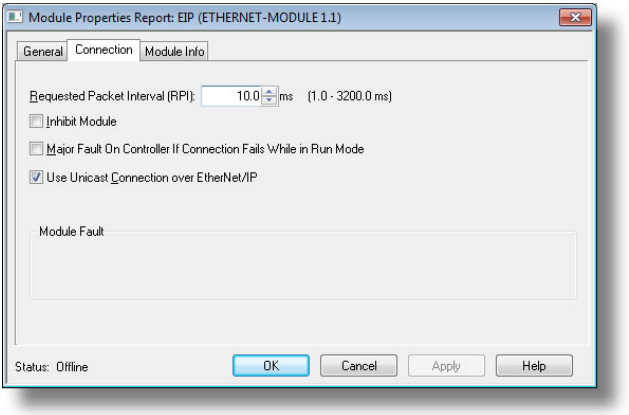


Figure 45: Module Properties Connection Tab

Confirm the setting of the Requested Packet Interval (RPI). The RPI defines the amount of time (in milliseconds) between data exchanges across an I/O connection. The smallest RPI supported by the interface card is 1ms. Click OK when done.

6) After adding the I/O Module to the configuration, the full I/O Configuration tree should appear similar to Figure 46.

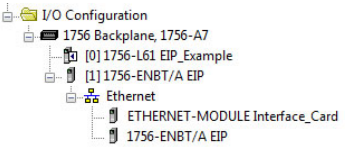


Figure 46: I/O Configuration Tree

7) Switch to online mode and download the project to the PLC. Verify that the newly-added inverter is available and operating correctly by observing any indications shown on the inverter's icon. When the inverter's icon is selected, its status and any available error messages will be displayed in the area below the project tree. Refer to Figure 47. Also confirm that the interface card's "Network Status" LED should be solid green, indicating an "online/connected" state.

8) By double-clicking "Controller Tags" in the project tree, it is possible to view the newly-added tags. Refer to Figure 48. The Interface_Card:C configuration tag is unused, the Interface_Card:I tag allows viewing of the input data, and the Interface_Card:O tag allows modification of the output data. These tags will be synchronized with the inverter at whatever rate was established for the module's RPI.

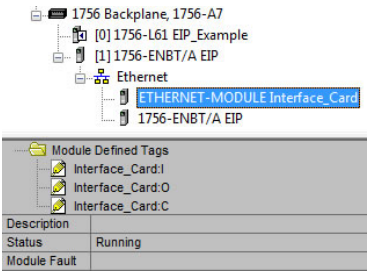


Figure 47: Online Module Status

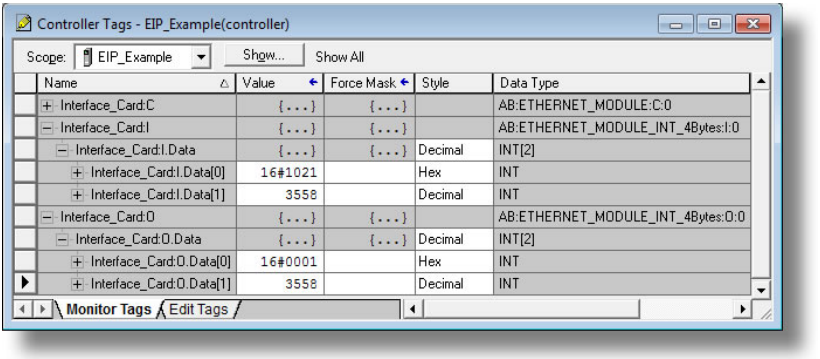


Figure 48: Controller Tags for I/O Access

We can directly interact with these tags in order to control and monitor the inverter. In Figure 48, for example, we can see that the first 16-bit word of output data (Interface_Card:O.Data[0]) has been set to a hexadecimal value of 0x0001. The default consumed data word configuration word offset 0 references function code S06, which is the inverter's command register. A value of 0x0001, therefore, means that the FWD (run forward) bit has been turned ON.

Similarly, we can see that the second 16-bit word of output data (Interface_Card:O.Data[1]) has been set to a decimal value of 3558. The default consumed data word configuration word offset 1 references function code S05, which is the inverter's frequency command register. A value of 3558, therefore, equates to a frequency command of 35.58Hz.

The input data from the inverter shows similar expected results. Values of 0x1021 and 3558 corresponding to M14 (status register) and M09 (output frequency), respectively, are consistent with the inverter running at the parameters commanded by the output tag.

8.2.11 ControlLogix Example: Generic Default I/O Add-On Instruction

The generic default I/O add-on instruction is a simple interface to command and monitor the inverter. It is based on the vendor-specific assembly instances 100 & 150 and the default produce and consume data configuration (refer to section 8.2.3). The add-on instruction is optional and provided for user convenience.

1) Complete all the steps in section 8.2.10.

2) Right click on "Add-On Instructions" in the controller organizer view and select "Import Add-On Instruction". Browse and import the generic default I/O add-on instruction. Refer to Figure 49.

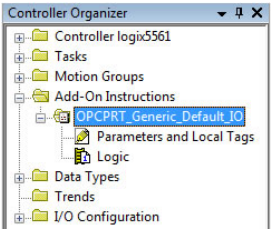


Figure 49: Generic Default IO Add-On Instruction

3) Double click "Controller Tags" in the controller organizer view and select the "Edit Tags" tab at the bottom.

4) Create the tags in Figure 50.

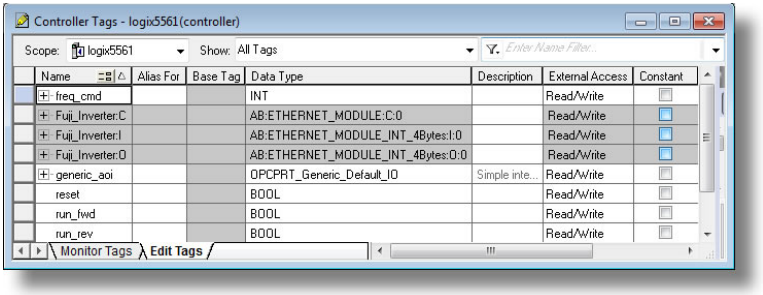


Figure 50: Create Generic Default AOI Tags

5) Double click “MainRoutine” under Tasks …MainTask …MainProgram in the controller organizer view.

6) Right click on the first ladder logic rung in the MainRoutine window and select “Add Ladder Element...”

7) The “Add Ladder Element” window appears.

8) Select the generic default I/O add-on instruction in the Add-On folder. Refer to Figure 51.

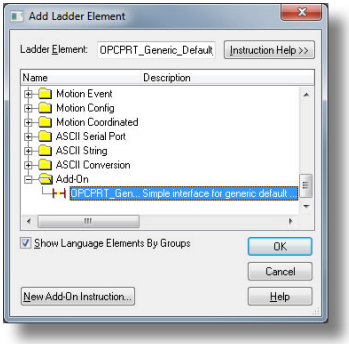


Figure 51: Add Generic Default Add-On Instruction

9) Click OK.

10) Edit the add-on instruction according to Figure 52.

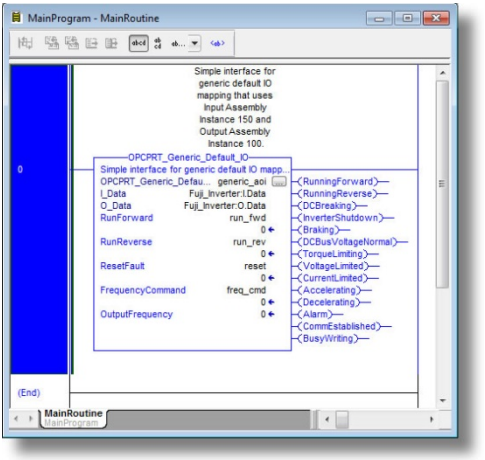


Figure 52: Configure Generic Default AOI

11) The program is now complete.

12) Save, download and run the program.

8.2.12 ControlLogix Example: AC/DC Drive Profile Add-On Instruction

The AC/DC drive profile add-on instruction is a simple interface to command and monitor the inverter. It is based on the assembly instances 21 & 71. The add-on instruction is optional and provided for user convenience.

1) Complete all the steps in section 8.2.10. Please note that the Assembly Input Instance must be changed to 71 and the Assembly Output Instance must be changed to 21. Refer to Figure 53.

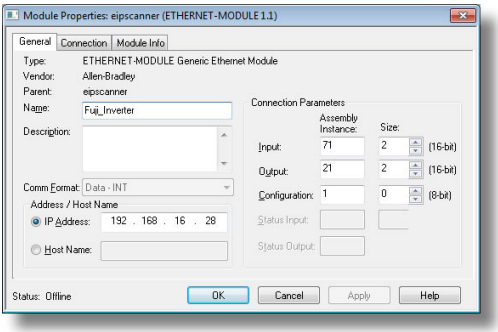


Figure 53: AC/DC Drive Profile Generic Ethernet Module Configuration

2) Right click on “Add-On Instructions” in the controller organizer view and select “Import Add-On Instruction”. Browse and import the AC/DC drive profile add-on instruction. Refer to Figure 54.

3) Double click “Controller Tags” in the controller organizer view and select the “Edit Tags” tab at the bottom.

4) Create the tags in Figure 55.

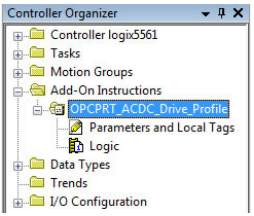


Figure 54: AC/DC Drive Profile Add-On Instruction

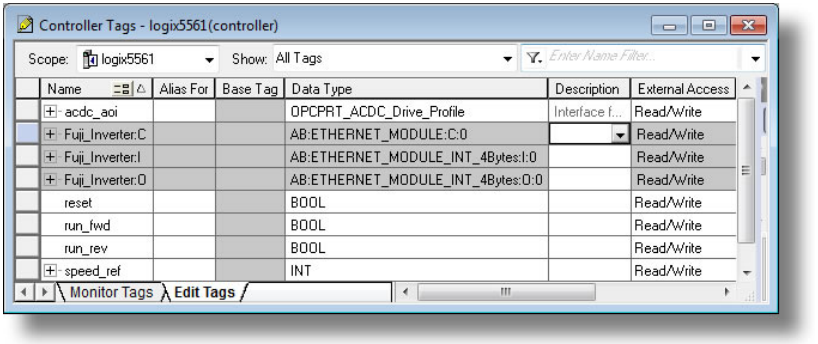


Figure 55: Create AC/DC Drive Profile AOI Tags

5) Double click “MainRoutine” under Tasks …MainTask …MainProgram in the controller organizer view.

6) Right click on the first ladder logic rung in the MainRoutine window and select “Add Ladder Element...”

7) The “Add Ladder Element” window appears.

8) Select the AC/DC drive profile add-on instruction in the Add-On folder. Refer to Figure 56.

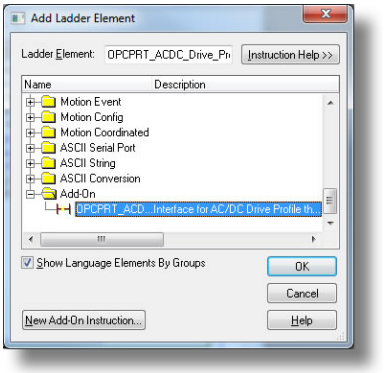


Figure 56: Add AC/DC Drive Profile Add-On Instruction

9) Click OK.

10) Edit the add-on instruction according to Figure 57.

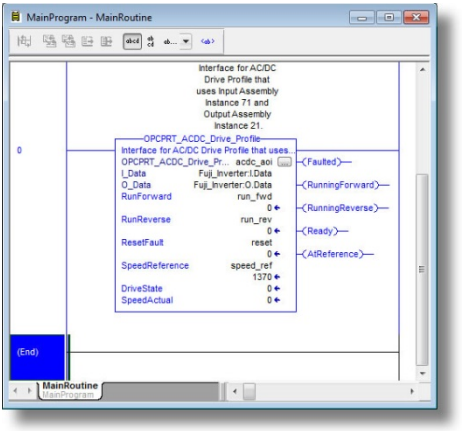


Figure 57: Configure AC/DC Drive Profile AOI

11) The program is now complete.

12) Save, download and run the program.

8.2.13 ControlLogix Example: Read a Block of Function Codes

This example program will show how to continuously read a block of function codes from the inverter with a single MSG instruction. Only one read request is outstanding at any given time.

1) Create new Tags.

a) Double click “Controller Tags” in the controller organizer view.

b) The “Controller Tags” window appears. Refer to Figure 58.

Figure 58: Create New Tags

c) Select the “Edit Tags” tab at the bottom.

d) Create a new tag by entering “connection” in the first blank Name field, and change its Data Type to “MESSAGE”. This tag will contain configuration information for the MSG instruction.

e) Select the “Monitor Tags” tab. Expand the “connection” tag by clicking on the “+” sign next to the tag name. Scroll down to the connection.UnconnectedTimeout field and change its value from the default 30000000 (30s in 1uS increments) to 1000000 (1s). This value determines how long to wait before timing out and retransmitting a connection request if a connection failure occurs.

f) Collapse the “connection” tag again by clicking on the “-“ sign next to the tag name.

g) Select the “Edit Tags” tab again. Create another new tag by entering “data_array” in the next blank Name field, and change its Data Type by typing in “INT[73]” in the Data Type field. This tag is an array of INTs that will be able to hold up to 73 16-bit function codes from the inverter. Always make sure that the destination tag size is large enough to hold all elements to be read.

2) Add a MSG instruction to the main program.

a) Double click “MainRoutine” under Tasks …MainTask …MainProgram in the controller organizer view.

b) Right click on the first ladder logic rung in the MainRoutine window and select “Add Ladder Element...”

c) The “Add Ladder Element” window appears.

d) Select the “MSG” instruction in the Input/Output folder. Refer to Figure 59.

e) Click OK.

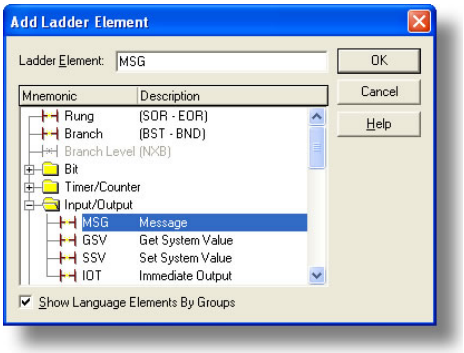


Figure 59: Adding a MSG Instruction

3) Add an XIO element to the main program.

a) Right click on the ladder logic rung containing the MSG instruction in the MainRoutine window and select “Add Ladder Element...” again.

b) The “Add Ladder Element” window appears.

c) Select the “XIO” element in the Bit folder. Refer to Figure 60.

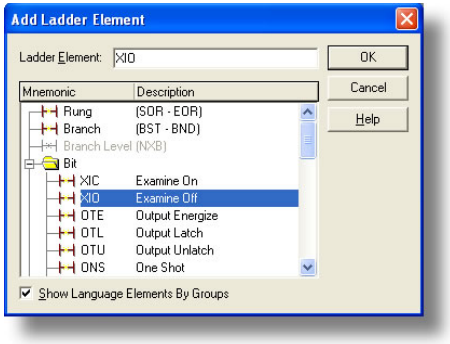


Figure 60: Adding an XIO Element

d) Click OK.

4) Configure the MSG instruction.

a) Edit the “Message Control” field on the MSG instruction to use the previously-created “connection” tag. Refer to Figure 61.

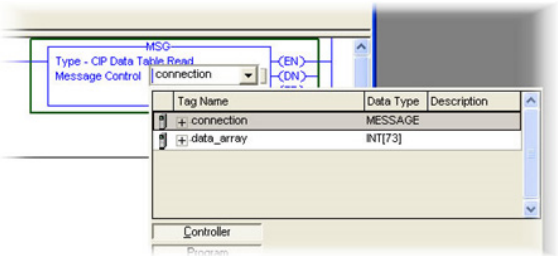


Figure 61: MSG Instruction Tag Assignment

b) Click the message configuration button (“…”) in the MSG instruction. The “Message Configuration” window will open. Refer to Figure 62.

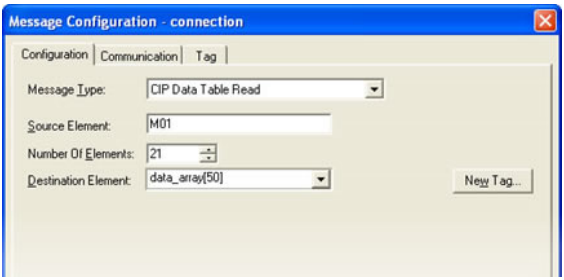


Figure 62: MSG Instruction Configuration

c) “Configuration” tab settings:

i) Change the “Message Type” to “CIP Data Table Read”.

ii) In the "Source Element” field, enter the read tag you wish to access (refer to section 8.2.11.) In this example, we will be reading a total of 21 function codes beginning at function code M01 (per-unit frequency reference – final command).

iii) Enter the Number Of Elements to read. In this example, we will read 21 function codes.

iv) For the Destination Element, select “data_array[50]”.

d) “Communication” tab settings (refer to Figure 63):

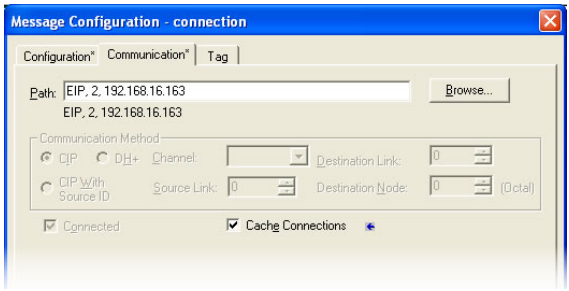


Figure 63: Setting the Communication Path

i) Enter the Path to the interface card. A typical path is formatted as “*Local_ENB,2,target_IP_address*”, where:

- *Local_ENB* is the name of the 1756-ENBx module in the local chassis (we named ours “EIP” in section 8.2.9),
- *2* is the Ethernet port of the 1756-ENBx module in the local chassis, and
- *target_IP_address* is the IP address of the target node.

In our example, this path would be entered as “EIP,2,192.168.16.163”.

ii) If “Cache Connections” is enabled (checked), the connection remains open after transmission. If disabled (unchecked), the connection is opened before and closed after every transmission. For efficiency, it is recommended to enable “Cache Connections”.

e) Click “OK” to close the MSG Configuration dialog. At this stage, MainRoutine should look like Figure 64.

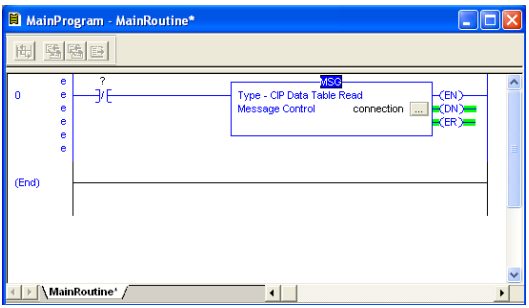


Figure 64: MainRoutine

5) Assign a tag to the XIO element.

a) Double-click on the XIO element located to the left of the MSG block. In the drop-down box, double-click on the “connection.EN” field. Refer to Figure 65. This configuration causes the MSG instruction to automatically retrigger itself when it completes. While this is acceptable for the purposes of this example, it can produce high network utilization. In actual practice, it may be desirable to incorporate additional logic elements to allow triggering the MSG instruction at a specific rate or under specific conditions.

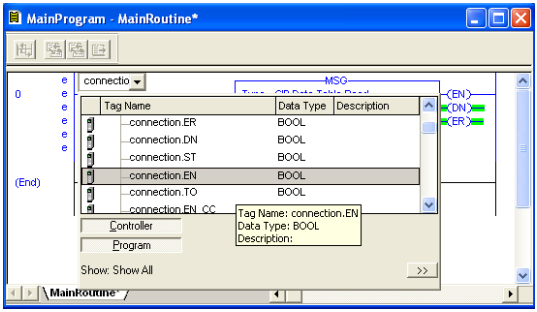


Figure 65: Configure XIO Element

6) The program is now complete. Refer to Figure 66.

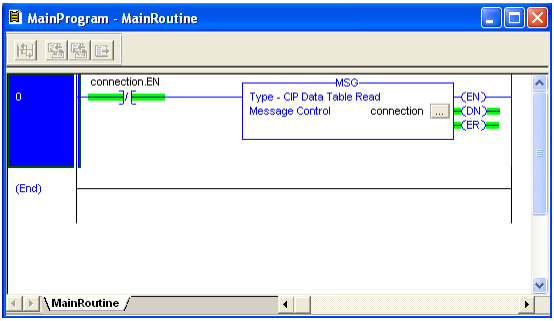


Figure 66: Complete Program

7) Save, download and run the program.

a) To view the values of the function codes being read from the interface card, double-click “Controller Tags” in the controller organizer view.

b) Select the “Monitor Tags” tab and expand the data_array tag.

c) 21 function code values starting at function code M01 are being continuously read from the interface card and placed in the 21 sequential offsets of data_array starting at the 50th offset (data_array[50]).

8.2.14 ControlLogix Example: Read a Single Function Code

The configuration and execution for reading a single function code is in general identical to that required for reading a block of function codes as detailed in section 8.2.13. The only difference is in the configuration of the MSG instruction. Figure 67 shows an example MSG instruction's Configuration tab, which will read a single tag (function code M14, the inverter's operation status register) and place it in the first element (offset 0) of data_array.

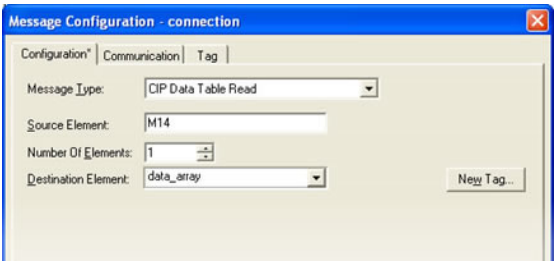


Figure 67: Read the Inverter's Status Function Code

8.2.15 ControlLogix Example: Multiple MSG Instructions

At times, reading from different groups of function codes may be necessary. For example, a specific application may require access to both configuration function codes and monitor function codes. To accomplish this task, multiple MSG instructions will need to be implemented in the PLC program.

The configuration and execution for implementing multiple MSG instructions is in general identical to that required for implementing just one MSG instruction. Each MSG instruction will require its own message controller tag. In the case of read MSG instructions, more than one instruction may use the same Destination Element tag, but the storage locations must not overlap. Figure 68 shows an example of two MSG instructions, each accessing different read tags. It is evident from this logic that "rd_connection" and "rd_connection2" are the two independent message controller tags created for these instructions.

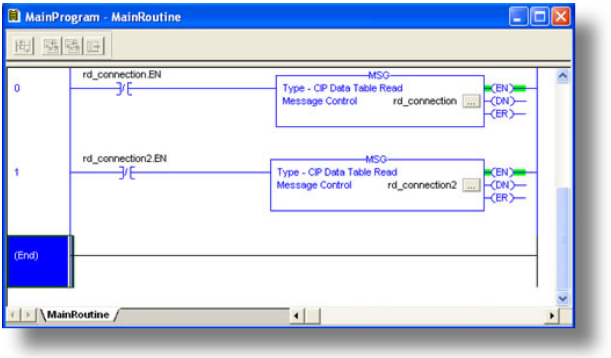


Figure 68: Reading Via Multiple MSG Instructions

8.2.16 ControlLogix Example: Reading and Writing

Often times, applications may need to both read data from and write data to the inverter. At a minimum, this will require two MSG instructions and two message controller tags. Figure 69 shows an example of three MSG instructions, one for reading and two for writing (the inverter's frequency command and command word). The only item of note that differentiates this example from the multiple-read example in section 8.2.15 is the addition of the en_xx_wr XIC elements. The reason for the addition of these elements is that while reading from a remote device is often continuously performed (monitoring), data is typically written to the remote device only when necessary (i.e. when the value to write has changed). This conserves both network bandwidth and potentially EEPROM lifespans on the target device. The en_xx_wr elements in this example, therefore, would typically be replaced in an actual application program by user-provided logic that controls the conditions under which write operations would be performed.

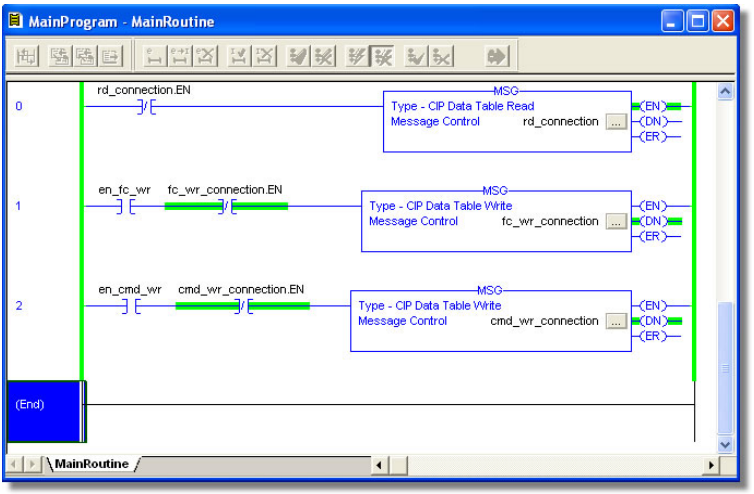


Figure 69: Reading and Writing via MSG Instructions

Figure 70 shows the configuration details of the example fc_wr_connection MSG instruction. Note that the chosen "Message Type" is "CIP Data Table Write", and that this instruction will only be writing to one inverter function code: namely, the frequency command (Destination Element is S05). The Source Element in this case is the 2nd element (starting from index 0) of an INT array tag named "wr_data".

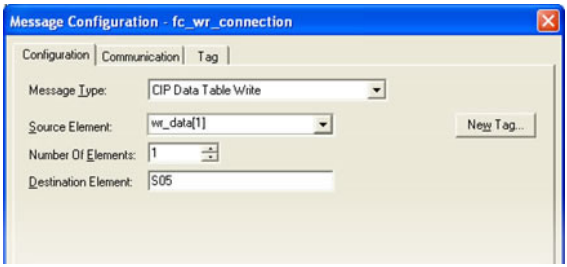


Figure 70: MSG Configuration for Writing

Note that when writing data via explicit messaging, use caution to ensure that the commanded function codes are not also simultaneously being commanded in the background via I/O messaging. Indeterminate behavior can occur if MSG instructions and background I/O data transfers are both writing to the same function codes. In other words, if the I/O messaging example procedure detailed in section 8.2.10 has already been implemented, and the same program is now being modified to implement explicit messaging, then it is recommended to inhibit the target module by selecting the "Inhibit Module" checkbox in the Connection tab of the Module Properties dialog.

8.3 Allen Bradley CSP (PCCC)

8.3.1 Overview

Ethernet-enabled Allen-Bradley legacy PLCs (such as the PLC5E, SLC-5/05, and MicroLogix series) use a protocol called CSP (Client Server Protocol) to communicate over the Ethernet network. The flavor of CSP used by these PLCs is also known as “PCCC” (Programmable Controller Communication Commands) and “AB Ethernet”. The interface card supports CSP for direct connectivity to these PLCs. Note that CSP runs under EtherNet/IP and is enabled by default when EtherNet/IP is added to the configuration.

If a connection timeout or socket-level error occurs, the driver will trigger a timeout event as described in section 5.4.1.

8.3.2 Explicit Messaging Via Typed Read/Write Services

Register (function code) contents are read from and written to the interface card via CSP by reference to an integer “file/section number” and an “offset/element” within that file. Reading is performed via the CSP “PLC5 Read” (DF1 protocol typed read, 0x68) service, and writing is performed via the CSP “PLC5 Write” (DF1 protocol typed write, 0x67) service. To read and write data, the client must reference a “target address” and the “size of elements”. The target address is constructed according to the conventions shown in section 8.3.3.

8.3.3 Inverter Function Code File Number Offset Format

The formula to calculate which register (function code) is targeted in the interface card is provided in Equation 7.

$$target\ register = (file\ number - 10) \times 100 + offset \qquad \textbf{Equation 7}$$

Refer to section 4.1 for converting function codes to register numbers. In Equation 7, “target register” $\in$[1…1899], “file number” $\in$[10…146] (which means N10…N146), and “offset” is restricted only by the limitations of the programming software (but is a value of 13668 max). Table 23 provides some examples of various combinations of file/section numbers and offsets/elements which can be used to access inverter function codes. Note that there are multiple different combinations of file/section numbers and offsets/elements that will result in the same inverter function code being accessed.

Table 23: CSP Target Register Examples

Function Code	Target Register	File/Section Number	Offset/Element	Address Format
F01	2	N10	1	N10:1
E05	262	N12	62	N12:62
S05	1798	N27	98	N27:98
d99	4964	N59	64	N59:64
J690	13659	N146	59	N146:59

In addition to providing access to the inverter function codes in their “standard” numerical locations as mentioned above, the function codes can also be accessed in a special “assembly object” type format by targeting integer file N50. What this means is that when N50 is targeted for reading, what is actually returned by the interface card is the user-defined function code data as ordered by the EtherNet/IP produced data word configuration (refer to section 8.2.3). Similarly, when N50 is targeted for writing, the written data is disseminated to the inverter’s function codes according to the definition contained in the EtherNet/IP consumed data word configuration. By appropriate configuration of the EtherNet/IP consumed and produced data word configuration, therefore, bulk access to non-contiguous but frequently-used inverter function codes can be conveniently provided by performing only one read and/or write instruction targeting file N50.

Because both the EtherNet/IP consumed and produced data word configurations are comprised of 32 function code definitions, the targeted “offset/element” must be within the range of 0 to 31 inclusive. Refer to Table 24 for some examples of N50 accesses.

Table 24: Examples of EtherNet/IP-Style Bulk Access via File N50

Offset/Element	Start Target Function Code of Configuration Array	Max Number of Accessible Elements
0	1st	32
:	:	:
15	16th	16
:	:	:
31	32nd	1

The application PLC program uses a MSG instruction that is configured with a “Data Table Address” from which to start the access and a “Size in Elements” which determines the number of items to access (read or write). The “Data Table Address” is constructed by selecting a “File/Section Number” and an “Offset/Element” according to Equation 7. For example, a “File/Section Number” of N27 and “Offset/Element” of 99 = N27:99, which corresponds to register 1799 (the inverter’s operation command register, function code S06).

8.3.4 SLC-5/05 Example: Read a Block of Function Codes

This example program will show how to continuously read a block of function codes from the inverter with a single MSG instruction. This action is performed via the Typed Read (a.k.a. “PLC5 Read”) message type. Only one read request is outstanding at any given time. Note that the steps for the MicroLogix and PLC5E may vary slightly, but in general are similar.

1) Run RSLogix 500, and create a new configuration.

2) Create a control and a data file.

a) Right click Data Files and select New… The “Create Data File” dialog box appears (refer to Figure 71).

b) To create a control file, enter a file number (e.g. 20), set the type to “Integer”, enter a descriptive name (e.g. “CONTROL”), and enter a number of elements (e.g. 100). Click OK to create the file. The control file is used to store configuration information pertaining to the functionality of the MSG instruction which will perform the data read.

c) Follow the same procedure to create a data file. This file will be used to store the incoming data read from the interface card. Enter a file number (e.g. 18), set the type to “Integer”, enter a descriptive name (e.g. “DATA”), and enter a number of elements (e.g. 200). Refer to Figure 72. Click OK to create the file.

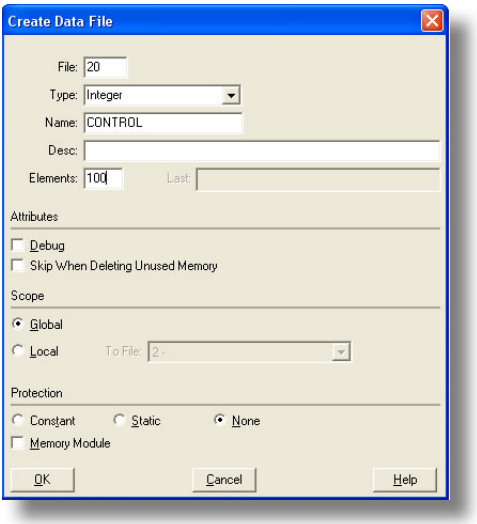


Figure 71: Creating a Control File

3) Add a MSG instruction to the program.

a) If not already visible, double-click “LAD2” under Project…Program Files in the controller organizer view to bring up the ladder logic program.

b) Right click on the default rung number on the left-hand side of the LAD2 window and select “Insert Rung”.

c) Right click on the rung number of the new editable rung and select “Append Instruction”.

d) Select the “MSG” instruction from the “Input/Output” classification, then click OK. Refer to Figure 73.

4) Add an XIO element to the program.

a) Right click on the rung number of the rung currently being edited and select “Append Instruction” again.

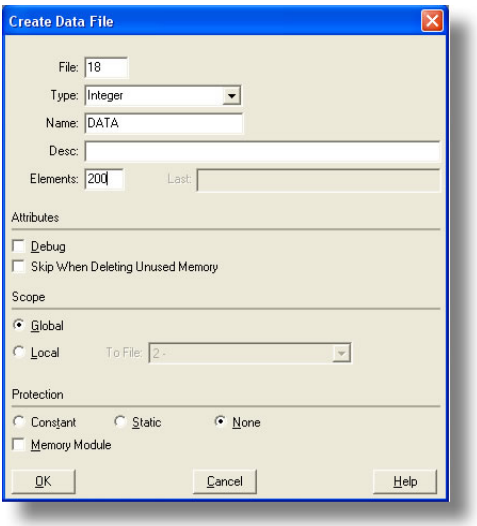


Figure 72: Creating a Data File

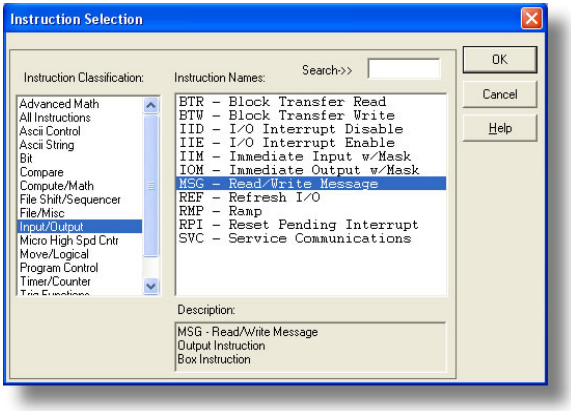


Figure 73: MSG Instruction Selection

b) Select the “XIO” instruction from the “Bit” classification, then click OK. Refer to Figure 74.

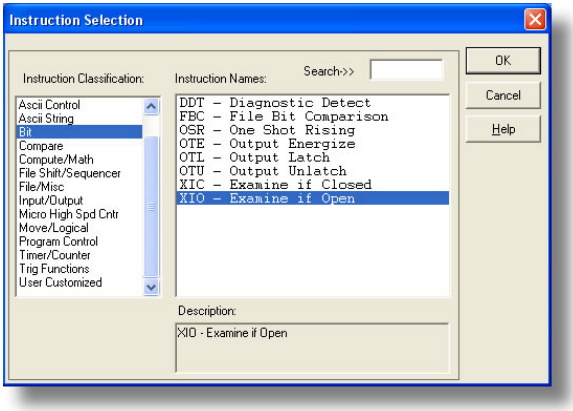


Figure 74: XIO Instruction Selection

5) Configure the MSG instruction.

a) Set the “Read/Write” field to “Read”, “Target Device” field to “PLC5”, “Local/Remote” field to “Local”, and “Control Block” to “N20:0”.

b) Upon hitting the <ENTER> key while in the “Control Block” entry box, the MSG Properties dialog box should appear (or it can be opened by clicking on the “Setup Screen” button at the bottom of the MSG instruction). Refer to Figure 75.

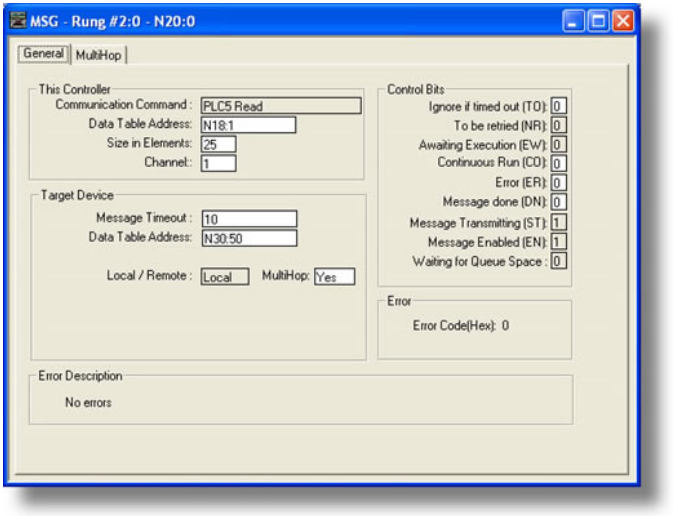


Figure 75: MSG Configuration, "General" Tab

c) In this example, we will be reading a total of 25 function codes beginning at N30:50 (register 2050 / function code M01). To configure this, under “This Controller” set the “Data Table Address” field to N18:1, set the “Size in Elements field” to 25, and set the “Channel” field to 1 (Ethernet).

d) Under “Target Device”, set the “Data Table Address” field to N30:50 (starting target register=2050) and set the “MultiHop” field to Yes to cause the “MultiHop” tab to appear.

e) Under the “MultiHop” tab settings, set the “To Address” in the first row to the inverter’s IP address, and the “To Address” in the second row to 0. Refer to Figure 76.

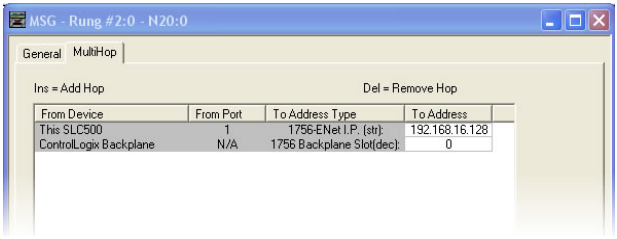


Figure 76: MSG Configuration, "MultiHop" Tab

f) Close the dialog box. At this point, the program should appear as shown in Figure 77.

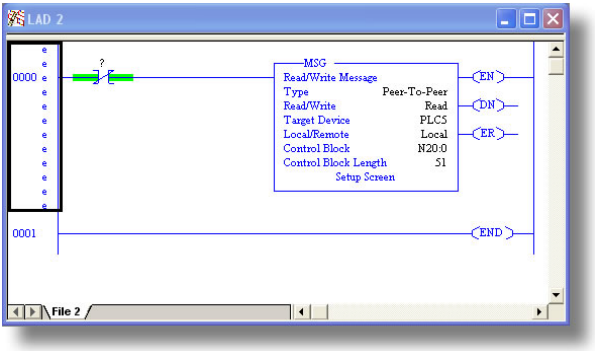


Figure 77: PLC Program after MSG Instruction Configuration

6) Assign a tag to the XIO element.

a) Double-click on the XIO element located to the left of the MSG block. Type in N20:0/15 (MSG instruction's enable bit). This configuration causes the MSG instruction to automatically retrigger itself when it completes. While this is acceptable for the purposes of this example, it can produce high network utilization. In actual practice, it may be desirable to incorporate additional logic elements to allow triggering the MSG instruction at a specific rate or under specific conditions.

7) The program is now complete. Refer to Figure 78.

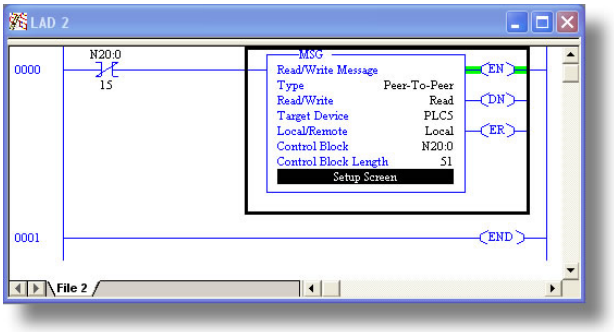


Figure 78: Completed PLC Program

8) Save, download, and run the program.

a) To view the function codes being read from the interface card, double-click the data file N18 under "Data Files" in the controller organizer view. 25 function code values starting at register #2050 are being continuously read from the interface card and placed in the 25 sequential offsets of N18 starting at N18:1. Refer to Figure 79. We can see that N18:9 (register 2058 / output frequency / function code M09) has a value of 2525 (25.25Hz), N18:12 (register 2061 / output voltage / function code M12) has a value of 610 (61.0V), etc.

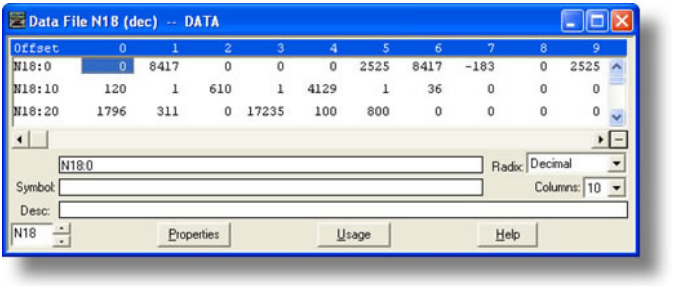


Figure 79: Monitoring the Data Being Read from the Inverter

8.3.5 SLC-5/05 Example: Read a Single Function Code

The configuration and execution for reading a single function code is in general identical to that required for reading a block of function codes as detailed in section 8.3.4. The only difference is in the configuration of the MSG instruction. Figure 80 shows an example MSG instruction's General tab, which will read a single element (N30:58, which corresponds to register 2058 / output frequency / function code M09) and place it in the first element (offset 0) of N18.

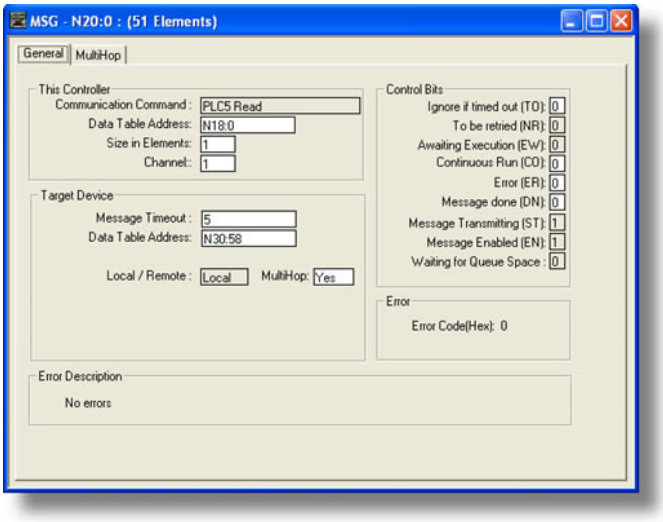


Figure 80: Read the Inverter's Status Function Code

8.3.6 SLC-5/05 Example: Multiple MSG Instructions

At times, reading from different groups of function codes may be necessary. For example, a specific application may require some function codes located in various disjoint locations in the register map. To accomplish this task efficiently, multiple MSG instructions can be implemented in the PLC program.

The configuration and execution for implementing multiple MSG instructions is in general identical to that required for implementing just one MSG instruction. Each MSG instruction will require its own message control file. In the case of read MSG instructions, more than one instruction may use the same data file to store the received function code values, but the storage locations must not overlap. Figure 81 shows an example of two MSG instructions, each accessing different target integer files. It is evident from this logic that N20 and N21 are the two independent message control files created for these instructions.

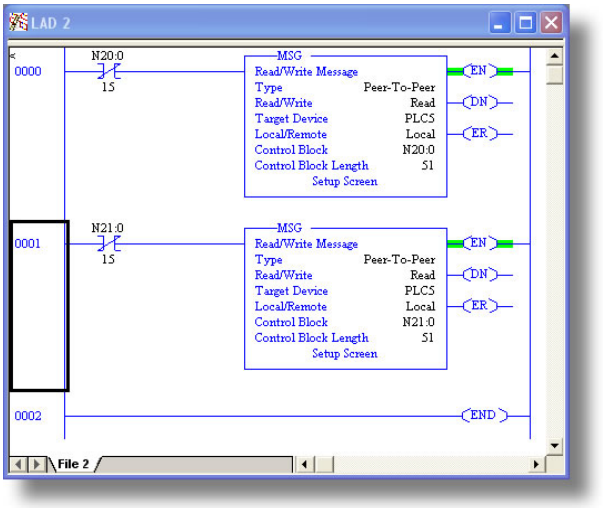


Figure 81: Multiple MSG Instructions

8.3.7 SLC-5/05 Example: Reading and Writing

Often times, applications may need to both read data from and write data to the inverter. At a minimum, this will require two MSG instructions and two message control files. Figure 82 shows an example of two MSG instructions, one for reading and one for writing. Note that the “Read/Write” field of each of the MSG instructions is set according to their function.

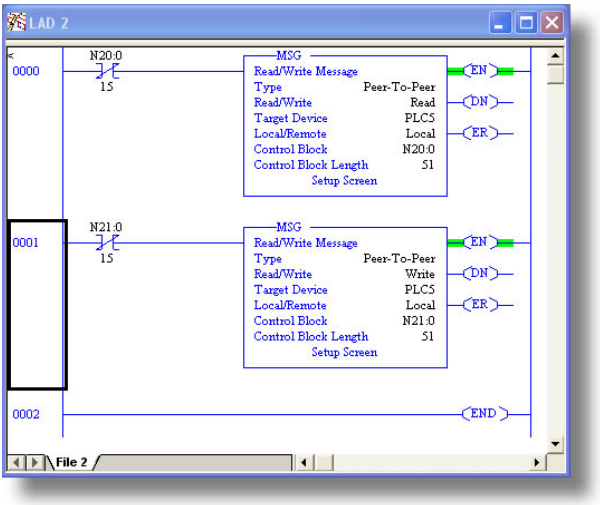


Figure 82: Reading and Writing via MSG Instructions

Figure 83 shows the configuration details of the “write” MSG instruction. Note that this instruction will only be writing to one inverter register: namely, register 1798 (function code S05 / frequency command). The source Data Table Address in this case is N18:30.

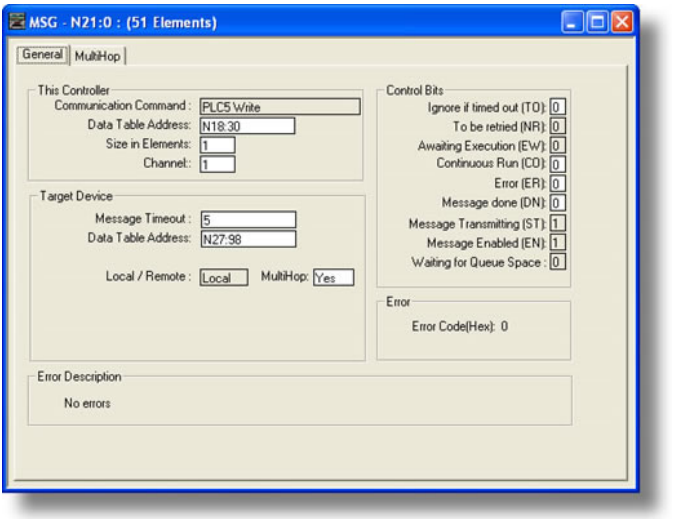


Figure 83: MSG Configuration for Writing

8.4 BACnet/IP

- The interface card supports the BACnet/IP (Annex J) protocol over Ethernet via a configurable UDP port (default value of 47808).
- The BACnet driver does not trigger timeout events (section 5.4.1).

8.4.1 Protocol Implementation Conformance Statement

BACnet Protocol

Date:	December 19, 2014
Vendor Name:	ICC, Inc.
Product Name:	Fuji FRENIC-Ace
Product Model Number:	OPC-PRT
Applications Software Version:	V1.1.0
Firmware Revision:	V1.1.0
BACnet Protocol Revision:	2

Product Description:

The Fuji Electric FRENIC series is a family of high-performance multifunctional inverters. Other features include ROHS compliance and built-in EMC filter.

BACnet Standard Device Profile (Annex L):

☐ BACnet Operator Workstation (B-OWS)
☐ BACnet Building Controller (B-BC)
☐ BACnet Advanced Application Controller (B-AAC)
☒ BACnet Application Specific Controller (B-ASC)
☐ BACnet Smart Sensor (B-SS)
☐ BACnet Smart Actuator (B-SA)

BACnet Interoperability Building Blocks Supported (Annex K):

☒ Data Sharing – ReadProperty-B (DS-RP-B)
☒ Data Sharing – ReadPropertyMultiple-B (DS-RPM-B)
☒ Data Sharing – WriteProperty-B (DS-WP-B)
☒ Device Management – Dynamic Device Binding-B (DM-DDB-B)
☒ Device Management – Dynamic Object Binding-B (DM-DOB-B)

Segmentation Capability:

None

☐ Segmented requests supported Window Size _____
☐ Segmented responses supported Window Size _____

Standard Object Types Supported:

See “Object Types/Property Support Table”.

Data Link Layer Options:

☒ BACnet IP, (Annex J)
☐ BACnet IP, (Annex J), Foreign Device
☐ ISO 8802-3, Ethernet (Clause 7)
☐ ANSI/ATA 878.1, 2.5 Mb. ARCNET (Clause 8)
☐ ANSI/ATA 878.1, RS-485 ARCNET (Clause 8), baud rate(s) _____
☐ MS/TP master (Clause 9), baud rate(s): 9600, 19200, 38400, 76800
☐ MS/TP slave (Clause 9), baud rate(s): _____
☐ Point-To-Point, EIA 232 (Clause 10), baud rate(s): _____
☐ Point-To-Point, modem, (Clause 10), baud rate(s): _____
☐ LonTalk, (Clause 11), medium: _____

☐ Other: _____

Device Address Binding:

Is static device binding supported? (This is currently for two-way communication with MS/TP slaves and certain other devise.) ☐ Yes ☒ No

Networking Options:

☐ Router, Clause 6 - List all routing configurations
☐ Annex H, BACnet Tunneling Router over IP
☐ BACnet/IP Broadcast Management Device (BBMD)
Does the BBMD support registrations by Foreign Devices? ☐ Yes ☐ No

Character Sets Supported:

Indicating support for multiple character sets does not imply that they can all be supported simultaneously.

☒ ANSI X3.4 ☐ IBM™/Microsoft™ DBCS ☐ ISO 8859-1
☐ ISO 10646 (UCS-2) ☐ ISO 10646 (UCS-4) ☐ JIS C 6226

If this product is a communication gateway, describe the types of non-BACnet equipment/networks(s) that the gateway supports: N/A

Datatypes Supported:

The following table summarizes the datatypes that are accepted (in the case of a write property service) and returned (in the case of a read property service) when targeting the present value property of each supported object type.

Object Type	Service	
	Read Property	Write Property
Analog Output	Real	Real, Unsigned, Integer, Null
Analog Input	Real	N/A
Analog Value	Real	Real, Unsigned, Integer, Null
Binary Output	Enumerated	Enumerated, Boolean, Real, Unsigned, Integer, Null
Binary Input	Enumerated	N/A
Binary Value	Enumerated	Enumerated, Boolean, Real, Unsigned, Integer, Null
Multi-state Output	Unsigned	Enumerated, Real, Unsigned, Integer, Null
Multi-state Input	Unsigned	N/A
Multi-state Value	Unsigned	Enumerated, Real, Unsigned, Integer, Null

Notes:
- The Null data type is used to relinquish a previously-commanded entry at the targeted priority in the priority array.

Object Types/Property Support Tables:

Table 25: BACnet Device Object Types /Properties Supported

Property	Object Type
	Device
Object Identifier	R
Object Name	R
Object Type	R
System Status	R
Vendor Name	R
Vendor Identifier	R
Model Name	R
Firmware Revision	R
Appl Software Revision	R
Protocol Version	R
Protocol Revision	R
Services Supported	R
Object Types Supported	R
Object List	R
Max APDU Length	R
Segmentation Support	R
APDU Timeout	R
Number APDU Retries	R
Device Address Binding	R
Database Revision	R

R – readable using BACnet services
W – readable and writable using BACnet services

Table 26: BACnet Binary Object Types /Properties Supported

Property	Object Type		
	Binary Input	Binary Output	Binary Value
Object Identifier	R	R	R
Object Name	R	R	R
Object Type	R	R	R
Present Value	R	W	W
Status Flags	R	R	R
Event State	R	R	R
Out-of-Service	R	R	R
Priority Array		R	R
Relinquish Default		R	R
Polarity	R	R	
Active Text	R	R	
Inactive Text	R	R	

R – readable using BACnet services
W – readable and writable using BACnet services

Table 27: BACnet Analog Object Types /Properties Supported

Property	Object Type		
	Analog Input	Analog Output	Analog Value
Object Identifier	R	R	R
Object Name	R	R	R
Object Type	R	R	R
Present Value	R	W	W
Status Flags	R	R	R
Event State	R	R	R
Out-of-Service	R	R	R
Units	R	R	R
Priority Array		R	R
Relinquish Default		R	R

R – readable using BACnet services
W – readable and writable using BACnet services

Table 28: BACnet Multi-state Object Types /Properties Supported

Property	Object Type		
	Multi-state Input	Multi-state Output	Multi-state Value
Object Identifier	R	R	R
Object Name	R	R	R
Object Type	R	R	R
Present Value	R	W	W
Status Flags	R	R	R
Event State	R	R	R
Out-of-Service	R	R	R
Number of States	R	R	R
Priority Array		R	R
Relinquish Default		R	R

R – readable using BACnet services
W – readable and writable using BACnet services

8.4.2 Default Supported Objects

This section will describe the default objects. Since the objects are configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration.

Table 29: Binary Input Object Instance Summary

Instance ID	Object Name	Description	Active/ Inactive Text
BI1	FWD_ROT_STATUS	Forward rotation status	forward/off
BI2	REV_ROT_STATUS	Reverse rotation status	reverse/off
BI3	EXT_STATUS	DC injection braking	braking/off
BI4	INVERTER_SHUTDOWN	Inverter shutdown	on/off
BI5	BRAKING	Braking	braking /off
BI6	NUV	DC bus voltage normal	on/off
BI7	TORQUE_LIMITING	Torque limited	on/off
BI8	VOLTAGE_LIMITING	Voltage limited	on/off
BI9	CURRENT_LIMITING	Current limited	on/off
BI10	ACCELERATING	Accelerating	on/off
BI11	DECELERATING	Decelerating	on/off
BI12	ALARM	Alarm	on/off
BI13	COMM_ESTABLISHED	Communications established	on/off
BI14	BUSY_WRITING	Busy writing	on/off

Table 30: Binary Output Object Instance Summary

Instance ID	Object Name	Description	Active/ Inactive Text
BO1	FWD_ROT_CMD	Forward rotation command	forward/off
BO2	REV_ROT_CMD	Reverse rotation command	reverse/off
BO3	X1	General purpose input	on/off
BO4	X2	General purpose input	on/off
BO5	X3	General purpose input	on/off
BO6	X4	General purpose input	on/off
BO7	X5	General purpose input	on/off
BO8	X6	General purpose input	on/off
BO9	X7	General purpose input	on/off
BO10	X8	General purpose input	on/off
BO11	X9	General purpose input	on/off
BO12	EN_TERMINAL	Enable terminal	on/off
BO13	XF_FWD	General purpose input	on/off
BO14	XR_REV	General purpose input	on/off
BO15	ALARM_RESET	Alarm reset	on/off

Table 31: Analog Input Object Instance Summary

Instance ID	Object Name	Description	Units
AI1	OUTPUT_FREQ	Output frequency	Hz
AI2	OUTPUT_CURRENT	Output current	Amps
AI3	OUTPUT_VOLTAGE	Output voltage	Volts
AI4	INPUT_POWER	Input power	kW
AI5	OUTPUT_POWER	Output power	kW

Table 32: Analog Output Object Instance Summary

Instance ID	Object Name	Description	Units
AO1	FREQ_REF	Frequency command	Hz
AO2	ACCEL_TIME	Acceleration time	Seconds
AO3	DECEL_TIME	Deceleration time	Seconds

8.4.3 Default Supported Object Details

This section will describe the default objects details. Since the objects are configurable, the system integrator is responsible for managing, maintaining, and documenting the actual configuration.

Binary Input Objects

BI1 ..... Indicates whether the inverter is running forward. Corresponds to function code M14, bit 0.
BI2 ..... Indicates whether the inverter is running reverse. Corresponds to function code M14, bit 1.
BI3 ..... Indicates DC injection braking or pre-exciting. Corresponds to function code M14, bit 2.
BI4 ..... Indicates inverter shutdown. Corresponds to function code M14, bit 3.
BI5 ..... Indicates braking. Corresponds to function code M14, bit 4.
BI6 ..... Indicates normal DC bus voltage. Corresponds to function code M14, bit 5.
BI7 ..... Indicates torque limited. Corresponds to function code M14, bit 6.
BI8 ..... Indicates voltage limited. Corresponds to function code M14, bit 7.
BI9 ..... Indicates current limited. Corresponds to function code M14, bit 8.
BI10 ..... Indicates acceleration. Corresponds to function code M14, bit 9.
BI11 ..... Indicates deceleration. Corresponds to function code M14, bit 10.
BI12 ..... Indicates alarm. Corresponds to function code M14, bit 11.
BI13 ..... Indicates communications established. Corresponds to function code M14, bit 12.
BI14 ..... Indicates function code write in progress. Corresponds to function code M14, bit 15.

Binary Output Objects

BO1 ..... Forward command. Corresponds to function code S06, bit 0.
BO2 ..... Reverse command. Corresponds to function code S06, bit 1.
BO3 ..... X1 command. Corresponds to function code S06, bit 2.
BO4 ..... X2 command. Corresponds to function code S06, bit 3.
BO5 ..... X3 command. Corresponds to function code S06, bit 4.
BO6 ..... X4 command. Corresponds to function code S06, bit 5.
BO7 ..... X5 command. Corresponds to function code S06, bit 6.
BO8 ..... X6 command. Corresponds to function code S06, bit 7.
BO9 ..... X7 command. Corresponds to function code S06, bit 8.
BO10 .... X8 command. Corresponds to function code S06, bit 9.
BO11 .... X9 command. Corresponds to function code S06, bit 10.
BO12 .... EN terminal command. Corresponds to function code S06, bit 11.
BO13 .... XF (FWD) command. Corresponds to function code S06, bit 13.
BO14 .... XR (REV) command. Corresponds to function code S06, bit 14.
BO15 .... Activates the alarm reset. Corresponds to function code S06, bit 15.

Analog Input Objects

AI1 ..... The output frequency of the inverter in 0.01 Hertz units (6000=60.00Hz). Corresponds to function code M09.
AI2 ..... The output current of the inverter in 0.1 or 0.01 Amp units (depends on inverter capacity). Corresponds to function code W05.
AI3 ..... The output voltage of the inverter in 0.1 Volt units (1000=100.0V). Corresponds to function code W06.
AI4 ..... Input power of the inverter in 0.01 kW units. Corresponds to function code W21.
AI5 ..... Output power of the inverter in 0.01 kW units. Corresponds to function code W22.

Analog Output Objects

AO1 ..... Frequency command of the inverter in 0.01 Hertz units. Corresponds to function code S05.
AO2 ..... Sets the acceleration time in 0.1 second units. Corresponds to function code S08.
AO3 ..... Sets the deceleration time in 0.1 second units. Corresponds to function code S09.

8.4.4 Server Settings

In the studio's Project panel, navigate to **OPC-PRT…Ethernet…BACnet/IP Server**.

UDP Port

This is the UDP port on which to transmit and receive BACnet/IP packets on the local subnet. The default value is 47808 (0xBAC0). To ensure successful communications, use caution when using a port setting other than the default value.

8.4.5 Node Settings

There are no node settings. A node is simply a container for objects.

8.4.6 Device Object Settings

A Device Object is automatically added to every node, and cannot be removed. The Device Object contains several configurable fields that must be appropriately set for each device residing on a BACnet network.

Device Name

Defines the node's name. The device name must be unique across the entire BACnet network. Enter a string of between 1 and 32 characters in length.

Instance Number

Defines the node's instance number. The instance number must be unique across the entire BACnet network. Enter a value between 0…4194302 (0x0…0x3FFFFE).

8.4.7 BACnet Object Settings

The BACnet server hosts BACnet objects which contain many different properties for any BACnet client on the network to access. The driver supports a variety of different BACnet objects. All supported properties of these objects are readable, while only the present value property is writable (for Outputs and Values only).

8.4.8 Analog Input Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed to 16-bit Unsigned.

Units

Select the desired units from this dropdown menu. If the desired units are not available in the dropdown menu, select "Other Units" and enter the appropriate enumerated value (as defined by the BACnet Specification) in the "Unit Value" field.

Unit Value

This field is enabled only when the "Units" selection is set to "Other Units". Enter the appropriate enumerated value (as defined by the BACnet Specification.)

8.4.9 Analog Output Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed to 16-bit Unsigned.

Units

Select the desired units from this dropdown menu. If the desired units are not available in the dropdown menu, select "Other Units" and enter the appropriate enumerated value (as defined by the BACnet Specification) in the "Unit Value" field.

Unit Value

This field is enabled only when the "Units" selection is set to "Other Units". Enter the appropriate enumerated value (as defined by the BACnet Specification.)

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are NULL.

8.4.10 Analog Value Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed to 16-bit Unsigned.

Units

Select the desired units from this dropdown menu. If the desired units are not available in the dropdown menu, select "Other Units" and enter the appropriate enumerated value (as defined by the BACnet Specification) in the "Unit Value" field.

Unit Value

This field is enabled only when the "Units" selection is set to "Other Units". Enter the appropriate enumerated value (as defined by the BACnet Specification.)

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are NULL.

8.4.11 Binary Input Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed to 16-bit Unsigned.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the “Function Code” that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one function code (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated function code.

The effect of the “Bitmask” field when reading: When the present value property of a binary object is read by a BACnet client, the bitmask is used to determine the active/inactive state of the object by inspecting the value in the designated function code at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated function code are set, then the object’s state will be returned as “active”. Else, the object’s state will be returned as “inactive”. This resultant state is reversed prior to being placed on the network if the object’s “Polarity” is set to “Reverse”.

Active Text

Specifies the description of the object’s “active” state. Enter a string of up to 32 characters in length. This field is optional and may be left blank.

Inactive Text

Specifies the description of the object’s “inactive” state. Enter a string of up to 32 characters in length. This field is optional and may be left blank.

Polarity

Indicates the relationship between the physical state of the object (as stored in the function code) and the logical state represented by the object’s present value property. If the physical state is active high, select “Normal” from this dropdown menu. If the physical state is active low, select “Reverse” from this dropdown menu. For further detail, refer to the “Bitmask” behavioral description.

8.4.12 Binary Output Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object’s instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object’s present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the “Function Code” that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one function code (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated function code.

The effect of the “Bitmask” field when writing: When the present value property of a binary object is set to “active” by a BACnet client, then the bit(s) in the designated function code indicated by the bitmask are set. Similarly, when the present value property of the object is set to “inactive”, then the bit(s) in the designated function code indicated by the bitmask are cleared. This setting/clearing behavior is reversed if the object’s “Polarity” is set to “Reverse”.

The effect of the “Bitmask” field when reading: When the present value property of a binary object is read by a BACnet client, the bitmask is used to determine the active/inactive state of the object by inspecting the value in the designated function code at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated function code are set, then the object’s state will be returned as “active”. Else, the object’s state will be returned as “inactive”. This resultant state is reversed prior to being placed on the network if the object’s “Polarity” is set to “Reverse”.

Active Text

Specifies the description of the object's "active" state. Enter a string of up to 32 characters in length. This field is optional and may be left blank.

Inactive Text

Specifies the description of the object's "inactive" state. Enter a string of up to 32 characters in length. This field is optional and may be left blank.

Polarity

Indicates the relationship between the physical state of the object (as stored in the function code) and the logical state represented by the object's present value property. If the physical state is active high, select "Normal" from this dropdown menu. If the physical state is active low, select "Reverse" from this dropdown menu. For further detail, refer to the "Bitmask" behavioral description.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are NULL.

8.4.13 Binary Value Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Bitmask

Specifies which bit(s) in the 16-bit value designated by the "Function Code" that the binary object will map to. This mechanism allows up to 16 binary objects to be simultaneously assigned to one function code (each binary object mapping to a single bit of that 16-bit word). It is possible to map binary objects to multiple bits within the designated function code.

The effect of the "Bitmask" field when writing: When the present value property of a binary object is set to "active" by a BACnet client, then the bit(s) in the designated function code indicated by the bitmask are set. Similarly, when the present value property of the object is set to "inactive", then the bit(s) in the designated function code indicated by the bitmask are cleared.

The effect of the "Bitmask" field when reading: When the present value property of a binary object is read by a BACnet client, the bitmask is used to determine the active/inactive state of the object by inspecting the value in the designated function code at the bit location(s) indicated in the bitmask. If all of the bit locations at the designated function code are set, then the object's state will be returned as "active". Else, the object's state will be returned as "inactive".

Active Text

Specifies the description of the object's "active" state. Enter a string of up to 32 characters in length. This field is optional and may be left blank.

Inactive Text

Specifies the description of the object's "inactive" state. Enter a string of up to 32 characters in length. This field is optional and may be left blank.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are NULL.

8.4.14 Multi-state Input Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

8.4.15 Multi-state Output Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are NULL.

8.4.16 Multi-state Value Object Settings

Object Name

The name of the BACnet object. Enter a string of between 1 and 32 characters in length. All object names must be unique within a node.

Instance

The BACnet object's instance number. Enter a value between 0…4194302 (0x0…0x3FFFFE).

Function Code

The inverter function code that the BACnet object's present value will access.

Data Type

Fixed at 16-Bit Unsigned.

Relinquish Default

Defines the default value to be used for an object's present value property when all entries in the object's priority array are NULL.

8.5 PROFINET IO

8.5.1 Overview

The PROFINET IO device driver allows a controller to interact with the interface card via cyclic data exchange and acyclic read/write requests. The I/O data is entirely user-configurable, and is utilized when a standard I/O module is chosen during network configuration.

Some other notes of interest are:

- Allows simultaneous access to only 1 PROFINET controller.
- Supports conformance class A and real time (RT) communication.
- Supports MRP (Media Redundancy Protocol) client.
- Supports DCP (Discovery Control Protocol).
- Supports alarms.
- Supports I&M.
- The lowest supported I/O Cycle Update Time (in STEP 7 or an equivalent hardware configuration tool) is 1ms.
- The GSDML file can be obtained from technical support.
- Supports several user-configurable I/O modules with up to 32 input words and 32 output words.
- Supports the PROFIdrive profile version 4.1.
- No explicit module selection is required on the interface card: the module will be selected automatically according to the controller's configuration.
- If a timeout occurs on the RT connection, the driver can be configured to trigger a timeout event as described in section 5.4.1. The timeout value is dictated by the PROFINET controller and is at least three times the IO Cycle update time. The timeout value is also known as the "IO Cycle Watchdog" time.

8.5.2 Device Settings

In the studio's **Project** panel, navigate to **OPC-PRT…Ethernet…PROFINET IO**.

Device Name

The device name / station name must be unique across the entire PROFINET network, because it is used by controllers to uniquely identify PROFINET devices. This string must conform to the device name requirements contained in the PROFINET specification.

8.5.3 Connection Timeout Options

In the studio's **Project** panel, navigate to **OPC-PRT…Ethernet…PROFINET IO**. The following configuration options will determine the actions to be taken by the card if the PROFINET IO connection is abnormally terminated or lost.

Timeout Action

Select an action from the drop down menu:

"None"..... No effect. The inverter will continue to operate with the last available settings.

"Apply Fail-safe Values"..... Apply the fail-safe values as described in section 5.4.1.

"Fault Drive"..... The behavior will depend on the timeout conditions set by the inverter (function codes o27 and o28), which may result in an Er5 fault. Refer to section 3.2.

Enable Drive Fault Reset

This will clear the Er5 fault once communication is re-established. This option is only available if the **Timeout Action** is set to "Fault Drive".

8.5.4 Cyclic I/O Produced and Consumed Data Access Settings

In the studio's **Project** panel, add **OPC-PRT…Ethernet…PROFINET IO…Produced Data Word** and/or **Consumed Data Word**.

The Produced Data Word and Consumed Data Word objects are only applicable when using the I/O module "IN: 32 WORDS, OUT: 32 WORDS", which is typically the case. The Produced Data Word

defines the structure of status data sent from the inverter to the controller. The Consumed Data Word objects will define the structure of the command data sent from the controller (for example, a Siemens PLC) to the inverter. These objects allow the creation of custom-built I/O data. Up to 32 “command” function code values can be sent to the inverter, and up to 32 “status” function code values can be sent back to the controller. Therefore, up to 32 Produced and 32 Consumed Data Word objects can be created. If a consumed word offset is not defined, that data will be ignored by the inverter. If a produce word offset is not defined, the value will default to 0. The size of the actual I/O produced and consumed data is determined by the PROFINET controller. The I/O data format is summarized in Table 33.

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Produced Data Word Offset

The value from the associated inverter function code will populate this word offset of the produced data that is to be sent to the controller. It is recommended to start at word offset 0.

Consumed Data Word Offset

The consumed data received from the controller at this word offset will contain the value to be written to the associated inverter function code. It is recommended to start at word offset 0.

Function Code

The inverter function code associated with the word offset. For the Produced Data Word object, enter a “status” function code to be monitored. For the Consumed Data Word object, enter a “command” function code that can be written.

Data Type

Each data word is fixed to 16-Bit Unsigned (equivalent to two bytes.) The data word is transferred in little endian format.

Table 33: User-Configurable Module I/O Data Format

Consumed Data (PLC to Inverter)		Produced Data (Inverter to PLC)	
Word Offset	**Function Code**	**Word Offset**	**Function Code**
0	Any	0	Any
1	Any	1	Any
…	Any	…	Any
30	Any	30	Any
31	Any	31	Any

The default I/O configuration is described in Table 34.

Note Always use the studio to confirm the configuration before commissioning the device.

Table 34: Default User-Configurable Module I/O Data Format

Consumed Data (PLC to Inverter)		Produced Data (Inverter to PLC)	
Word Offset	**Function Code**	**Word Offset**	**Function Code**
0	S06	0	M14
1	S05	1	M09
…	None	…	None
30	None	30	None
31	None	31	None

8.5.5 PROFIdrive Profile

For optimal interoperability, the interface card supports the PROFIdrive profile version 4.1. No special configuration of the interface card is required when using the PROFIdrive profile. The controller **must** support the PROFIdrive profile and **must** be configured to use the “Standard Telegram 1” module on the interface card. If the controller does not support the PROFIdrive profile, use the configurable I/O “IN: 32

WORDS, OUT: 32 WORDS" module. The PROFIdrive profile is only partially described in this manual due to its complexity. The complete PROFIdrive profile specifications can be obtained from http://www.profibus.com/.

Some other notes of interest include:

- Implements Application Class 1 (standard drive)
- Supports only Standard Telegram 1 (ST1, PZD-2/2) on slot 1 (similar to Profibus PPO type 3)
- Supports only Speed Control Mode

8.5.5.1 PROFIdrive standard telegram 1

The standard telegram 1 mapping is described in Table 35.

Table 35: Standard Telegram 1

IO Data Word Offset	Setpoint (PLC to Inverter)		Actual Value (Inverter to PLC)	
	Significance	Description	Significance	Description
0	STW1	Control word 1	ZSW1	Status word 1
1	NSOLL_A	Reference speed setpoint	NIST_A	Speed actual

8.5.5.2 PROFIdrive control and status words

The control word, STW1, is the principal means for controlling the drive. It is sent by the controller (PLC) to the device (inverter). The bitmapping for the control word is described in Table 36. The status word, ZSW1, returns status information from the inverter to the controller. The bitmapping for the status word is described in Table 37.

Table 36: STW1 Control Word Mapping

Bit	Value	Significance	Description
0	1	ON	Run command ON
	0	OFF	Run command OFF
1	1	ON2	No coast stop
	0	OFF2	Coast to a stop
2	1	ON3	No quick stop
	0	OFF3	Quick stop
3	1	Enable Operation	Enable inverter operation
	0	Disable Operation	Disable inverter operation
4	1	Enable Ramp Generator	Enable the ramp frequency generator (RFG)
	0	Disable Ramp Generator	Hold the output frequency to 0 Hz
5	1	Unfreeze Ramp Generator	Unfreeze the RFG
	0	Freeze Ramp Generator	Freeze the RFG with the current output frequency
6	1	Enable Setpoint	Enable command
	0	Disable Setpoint	Disable command
7	1	Fault Acknowledge	Reset the alarm on a positive edge (0→1 transition)
	0	No significance	Do not reset the alarm
8 - 9	Not used	---	---
10	1	Control By PLC	Enable remote control. The IO process data is valid.
	0	No Control By PLC	Disable remote control. The IO process data is not valid.
11 - 15	Not used	---	---

Table 37: ZSW1 Status Word Mapping

Bit	Value	Significance	Description
0	1	Ready To Switch ON	Ready to run command ON
	0	Not Ready To Switch ON	Not ready to run command ON
1	1	Ready to Operate	Ready to run
	0	Not Ready To Operate	Not ready to run

Bit	Value	Significance	Description
2	1	Operation Enabled	Running
	0	Operation Disabled	Running disabled
3	1	Fault Present	Inverter tripped as indicated by ALM. Refer to function code M14 bit 11.
	0	No Fault	No trip present as indicated by ALM. Refer to function code M14 bit 11.
4	1	Coast Stop Not Activated	Follows STW1 bit 1, ON2 active
	0	Coast Stop Activated	Follows STW1 bit 1, OFF2 active
5	1	Quick Stop Not Activated	Follows STW1 bit 2, ON3 active
	0	Quick Stop Activated	Follows STW1 bit 2, OFF3 active
6	1	Switch ON Inhibited	Not ready to run command ON
	0	Switch ON Not Inhibited	Ready to run command ON
7	Not Used	---	---
8	1	Speed Within Tolerance	Actual value equals the reference value and is within the tolerance as indicated by FAR. Refer to function codes M70 bit 1 and E30.
	0	Speed Out Of Tolerance	Actual value differs from the reference value or is outside of the tolerance as indicated by FAR. Refer to function codes M70 bit 1 and E30.
9	1	Control Requested	Control by PLC is enabled as indicated by RL. Refer to function code M14 bit 12.
	0	No Control Requested	Control is not possible by the controller as indicated by RL. Refer to function code M14 bit 12.
10	1	Frequency Reached Or Exceeded	The actual value ≥ max reference value as indicated by FDT. Refer to function codes M70 bit 2 and E31.
	0	Frequency Not Reached	The actual value < max reference value as indicated by FDT. Refer to function codes M70 bit 2 and E31.
11 - 15	Not used	---	---

8.5.5.3 PROFIdrive reference speed setpoint and actual speed

The speed setpoint value, NSOLL_A, is the commanded speed reference (normalized) sent from the controller to the inverter. Similarly, the speed actual value, NIST_A, is the actual operating speed (normalized) of the inverter sent back to the controller. As the inverter natively operates in units of Hz, the following conversion equations are applied within the interface card:

NSOLL_A: The inverter reference speed setpoint is a normalized value. The interface card applies the Normalize-to-Hz conversion indicated in Equation 8 in order to determine the appropriate frequency command value (in units of Hz) to be written to function code S05 (frequency command).

$$Hz = \frac{NSOLL\_A \times Max\ Frequency}{0x4000}$$ **Equation 8**

NIST_A: The inverter operating actual speed is a normalized value that is calculated from inverter function code M09 (output frequency). The interface card applies the Hz-to-Normalize conversion indicated in Equation 9 in order to determine the appropriate operating speed actual (normalized).

$$NIST\_A = \frac{Hz \times 0x4000}{Max\ Frequency}$$ **Equation 9**

The “Max Frequency” term which appears in Equation 8 and Equation 9 is obtained from the setting of inverter function code F03 (maximum frequency 1).

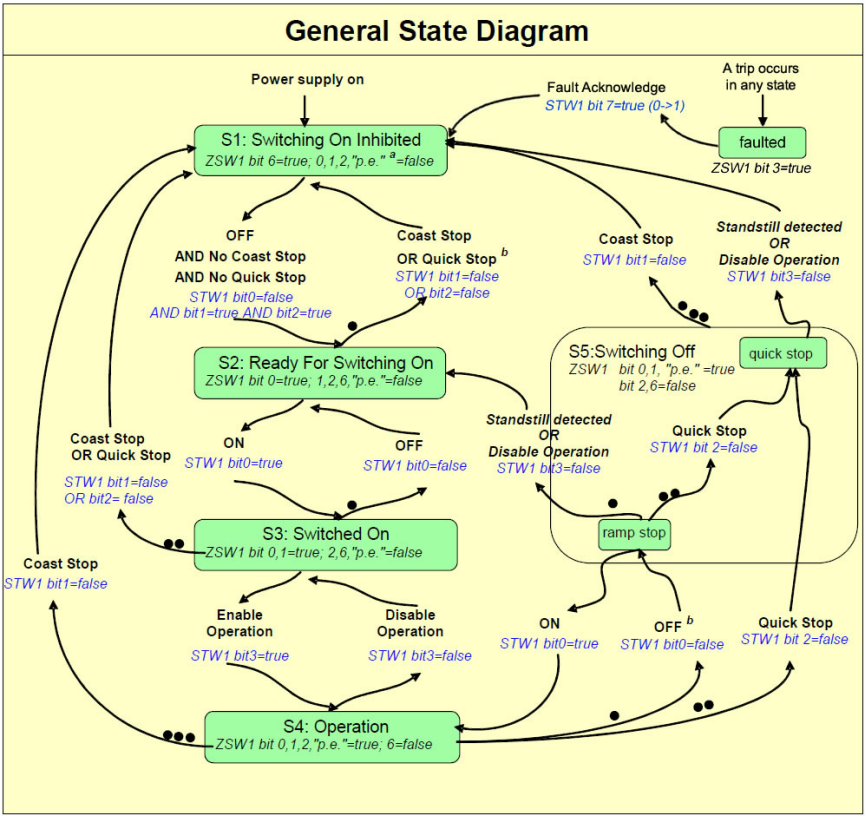
A normalized value of 0x4000 corresponds to 100% of the maximum frequency. A positive normalized value indicates forward rotation and a negative normalized value indicates reverse rotation.

Note The value of F03 is read by the interface card only at boot-up. If the value of this function code is changed, then the interface card must be rebooted in order for it to read the new value from the inverter.

8.5.5.4 PROFIdrive state diagram

The state diagram is displayed in Figure 84.

Figure 84: PROFIdrive State Diagram



8.5.5.5 PROFIdrive-specific parameters

The PROFIdrive-specific parameters are shown in Table 38. The parameters are read-only.

Table 38: PROFIdrive-Specific Parameters

PNU	Index	Description
711	None	NSOLL_A – Speed setpoint A
712	None	NIST_A – Speed actual A
833	None	STW1 – Control word 1
834	None	ZSW1 – Status word 1
922	None	Telegram selection = 1 (Standard telegram 1)
923	1,2,5,6	List of all parameters for signals
944	None	Fault message counter
947	0..3	Fault number (M16...M19)
964	0..6	Drive Unit identification
965	None	Profile identification number = Profile 3, Version 4.1
975	0..7	DO identification
980	0..5	Number list of defined parameter
1401	None	DO IO Data reference parameter

8.5.6 Acyclic Data Access

Any inverter function code can be accessed via PROFINET acyclic services. To accomplish this, set the API to 0, Slot to 1 and SubSlot to 1. The record number/index value is equivalent to the desired function code register number described in section 4.1. The length is specified according to the number of bytes to access. Since each register corresponds to 2 bytes of data, the length must be an even number.

8.5.7 STEP 7 Hardware Configuration Example

The following example will use STEP 7 to demonstrate the basic and typical hardware configuration procedure, which will apply to similar configuration software. The example will not cover all features of STEP 7. Any questions regarding STEP 7 (or similar configuration software) must be directed at the vendor of the software.

8.5.7.1 Register the GSDML file

Navigate to **Options…Install GSD File…** as shown in Figure 85.

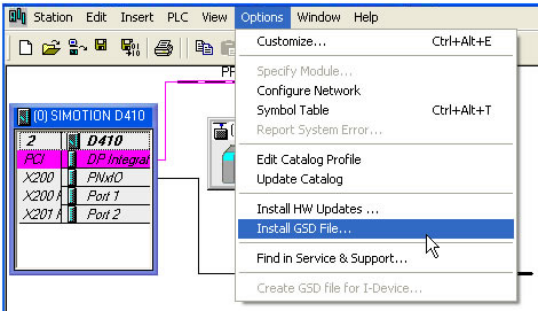


Figure 85: Install GSD File Menu Option

Locate and install the GSDML file as shown in Figure 86.

Figure 86: Successfully Installed GSDML File

Confirm that the device has been added to the catalog or device tree as shown in Figure 87.

Figure 87: Updated GSDML Device Tree

8.5.7.2 Add the device to the configuration

Select the device in the device tree and drag the device onto the PROFINET IO system in the configuration as shown in Figure 88.

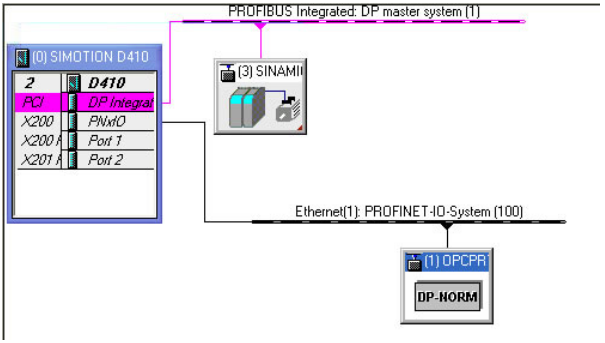


Figure 88: Add Device to Configuration

8.5.7.3 Configure the device properties

Open the device properties and assign a **unique** Device name as shown in Figure 89.

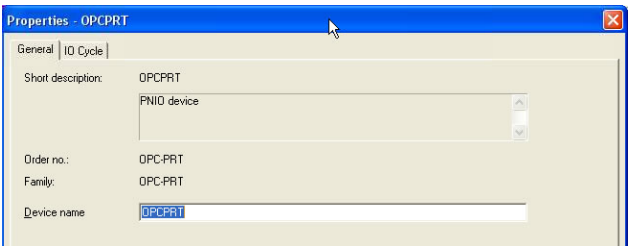


Figure 89: Assign Unique Device Name

Assign a **unique** and **compatible** IP address for your target network as shown in Figure 90.

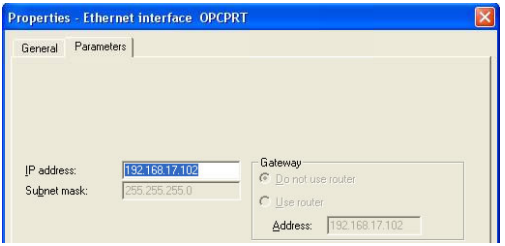


Figure 90: Assign Unique Compatible IP Address

Set the I/O Cycle Update Time as shown in Figure 91.

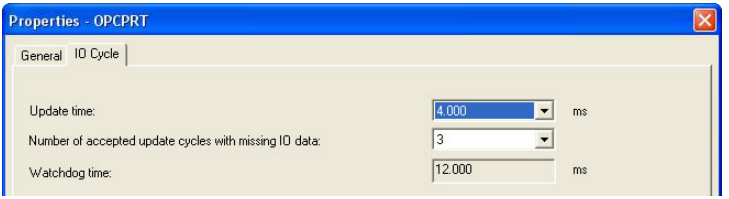


Figure 91: Set I/O Cycle Update Time

8.5.7.4 Assign the I/O module

In the device tree, expand the OPC-PRT module to view the available Virtual IO modules as shown in Figure 87. By default, there are two modules:

1) IN: 32 WORDS, OUT: 32 WORDS
    a. User-configurable (refer to section 8.5.3)
2) Standard Telegram 1
    a. PROFIdrive profile (refer to section 8.5.5)

First, select the OPC-PRT node in the configuration. Next, select an OPC-PRT Virtual IO module from the device tree and drag it into Slot 1 of the OPC-PRT. In this example, the “IN: 32 WORDS, OUT: 32 WORDS” module was added to Slot 1 in the configuration as shown in Figure 92.

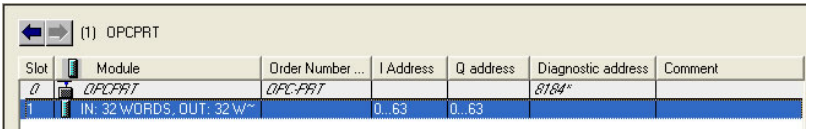


Figure 92: Add Module to Slot

8.5.7.5 Online device discovery and configuration

Navigate to **PLC…Ethernet…Edit Ethernet Node**. Click **Browse** to discover and view the online PROFINET devices on the network as shown in Figure 93.

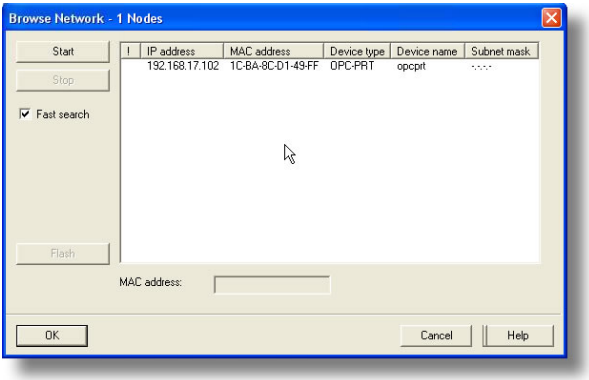


Figure 93: Discover PROFINET Devices on the Network

If the **Device name** and **IP address** do not match the values set in the configuration, select the device and click **OK**. Any non-matching value must be assigned to the device as shown in Figure 94.

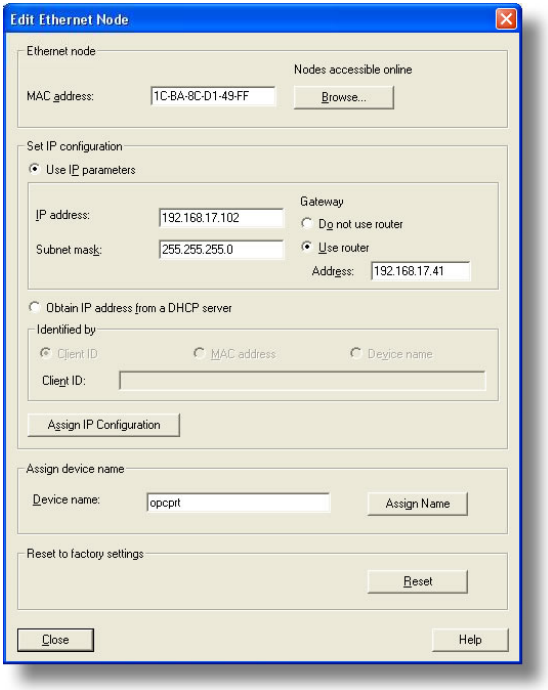


Figure 94: Configure Online Device

8.5.7.6 Save the configuration

The hardware configuration is now complete. Save and perform any necessary compilation of the configuration. The PLC application program can then be started. Please consult with the vendor of your PROFINET PLC software for additional configuration details.

9 TROUBLESHOOTING

Although by no means exhaustive, Table 39 provides possible causes behind some of the most common errors experienced when using the interface card.

Table 39: Troubleshooting

Problem	Symptom	Solution
No communications between the interface card and the inverter	Inverter displays "Err4" code	• Confirm that the interface card connector is properly seated. • Rebooting the interface card via the Fuji Configuration Studio disrupts the communication with the inverter. Reset the fault. • If the card is connected in a ring topology, the ring must support MRP. Otherwise a ring topology will result in an Ethernet loop.
No communications between the network and the interface card	Communications cannot be established, the Ethernet "link" LED is off, or the Ethernet "activity" LED flashes only infrequently or not at all	• Confirm that the card is running normally and connected to the local Ethernet network. • Ensure that the card is programmed with compatible network settings. Consult with your network administrator to determine the compatible settings. • Confirm that the destination IP address programmed into the controller equipment or computer matches that of the interface card, as displayed by the studio. • Confirm that intermediate firewalls or routers have been configured to allow access to the interface via the applicable TCP/UDP ports. • Try a known working Ethernet cable and switch. • If attempting to access the web server on a computer whose web browser is configured to use a proxy server, ensure that the proxy server is accessible to the computer, and that the interface card is accessible to the proxy server.
No PROFINET communication	PROFINET I/O communication cannot be established. The "Network Status" LED is not solid green.	• Confirm that the card's PROFINET device name matches the name assigned in the controller's configuration. • Confirm that the card's network settings match the settings assigned in the controller's configuration. • Confirm that the I/O cycle update time is set to 1ms or larger. • Ensure that the card is connected to a 100Mbps full duplex capable switch. • Ensure that the card can be discovered using the controller's discovery tool.
Unable to control the inverter via network communications	Writing to command and frequency function codes/registers has no apparent effect on inverter operation	• Confirm that the applicable inverter function codes are set to allow network control (refer to section 3.1). • If using the inverter's terminal contacts, refer to the inverter's instruction manual to determine the appropriate behavior and priority.

Problem	Symptom	Solution
XML socket connection failed	Message on a web server tab information window	TCP port 843 is blocked by a firewall, router or some other intermediate network equipment.
New web server content not loading after web server update	Old web server content is displayed	The internet browser has cached the old web server content. Clear the internet browser's cache before attempting to load the new web server content.
Web page does not display properly	Corrupt web server or outdated flash player plugin	• Ensure that USB and FTP are disconnected. • Download and install the latest flash player plugin from Adobe. • Delete the "WEB" folder from the card's file system and copy a valid default "WEB" folder to the card's file system.
Studio cannot discover the card	The studio does not display the card under "Online Devices"	• Confirm that the card is running normally and connected via USB or to the local Ethernet network. • Confirm that the module and network status LEDs blink the green/red startup sequence when power is first applied. • Add the studio as an exception to the computer's firewall. • Add UDP port 4334 as an exception to the firewall. • Temporarily disable the computer's firewall.
Studio cannot access file system	The studio displays an error when uploading and downloading the configuration.	If the studio continually displays an error regarding access to the file system, the file system may be corrupt. Please format the card's file system and then restore the configuration.
Firmware-generated error	"MODULE STATUS" LED is flashing red. The number of times the LED flashes indicates an error code.	Record the error code blinking pattern and contact technical support for further assistance.

47520 Westinghouse Dr.
Fremont, CA 94539
Tel: 510.440.1060
Fax: 510.440.1063
http://www.americas.fujielectric.com